VLBA Technical Report No. 44
THE VLBA CORRELATOR
FFT CONTROL CARD (FCC)
AND
FFT CARD MANUAL
VOLUME 1 of 2

fftmanv1.doc Microsoft Word not under SCCS control

Master file stored in /corrdwgs/manuals/fftman/SCCS
(where the SCCS file would have been stored)

This page intentionally left blank

TABLE OF CONTENTS

(VOLUME I)

Table of Figures

This page intentionally left blank.

1.0 Introduction: Function of FCC and FFT Cards

The FFT cards receive the reconstructed data streams from the Playback Interfaces (PBIs). The data is fringe derotated. FFTs are performed on the data, as the first part of an FX correlation process. Fractional Sample Time Corrections (FSTC) are made on the data. The data is passed on to the Multiplier Accumulate (MAC) cards. The FFT Control Card (FCC) provides the control signals required by the FFT Cards.

Each rack has an FFT Bin which contains one FCC and ten FFT cards. They are distributed as follows:

	Rack 0	Rack 1	Rack 2	Rack 3
Stations	0-9	10-19	0-9	10-19
Channels	0-3	0-3	4-7	4-7

One FFT card contains four FFT pipelines, 0-3. For FFT Sizes <= 512 input points, the pipelines can handle the data from four baseband channels. For FFT Size = 1K, two pairs of pipelines can handle the data from two baseband channels. For FFT Size = 2K, all four pipelines are used to handle the data from a single baseband channel.

2.0 The FFT Control Card (FCC)

2.1 Overview of the FCC

2.1.1 Introduction

2.1.1.1 FCC Functions

The function of the FFT Control Card (FCC) is to provide timing and control signals to one bins worth of FFT cards. A summary of the functions provided by the FCC card is given below. See k046d01.blk, the FCC Block Diagram, in Volume II.

1) Provide trig table values to the FFT engine butterfly stages on the FFT cards.

2) Perform hardware model computations every fringe cycle (4.128 msec).

2a) Send the resulting 2-coefficient fringe model parameters to their respective FFT cards.

2b) Send the integer part of the resulting delay model values to the respective DEF cards.

2c) Keep the resulting fractional delay coefficients and use them to perform FFT cycle computations for the fractional delay correction.

3) Provide the pulsar gate generator.

4) Provide an FFT cycle sequencer and time all of the control signals correctly within the FFT and fringe cycles.

5) Provide an HCB communication link with the real time computer system (RTS).

2.1.1.2 The FFT Control Card (FCC) Files

The FCC Orcad files are stored in /home/azalea/corrdwgs/fcc/sch/SCCS as follows:

The schematic for the FCC is given in drawing 1013d0x.sch, entered from 1013d01.sch. (See Volume II)

The layout of the FCC is in z006d01.lay. This has the official chip types for each position. (See Volume II)

The chip type in each of the 5 FCCs is listed in fcc_ic.lst stored with the schematic files. (Note all five FCCs do not have the "official" chip types installed.)

The FSTC Xilinx block diagrams are in k027*.blk, pointed to by k027top.blk. (See Volume II for sheets 2 and 3)

The pal schematics are in pal*.sch, pointed to by fcctop.pal. (See Volume II)

The pal Abel and jed files are stored in vlbsoft/pal_prom/pals/fcc/SCCS.

The FCC microprocessor source code is stored in vlbsoft/fccasm/SCCS.

The hcb protocol is in vlbsoft/fccasm/d026fcc.hcb under SCCS. This describes the Real Time System(RTS) commands for interfacing with the FCC microprocessor. (See Appendix I)

The microcode for the floating point chip is in vlbsoft/pal_prom/proms/fccfpc/SCCS. See the readme.txt in that area for details.

The code for the fcc sequencer is in vlbsoft/pal_prom/proms/fccseq/SCCS.

Trig tables are in vlbsoft/fftangles/SCCS.

2.1.2 FCC Functional Sections

2.1.2.1 The FFT Cycle Sequencer

The FFT cycle sequencer of the FCC is seen on sheet 2 of the FCC schematic in the upper left hand corner. (See Volume II). This logic provides a 516-state sequencer which is synchronized by the Master Control Card (MCC) ECL_FFTINIT signals. The 74F163 counters (14E, 15E, and 16E) count through the 516 states of the sequencer to address 516 memory locations in the four 7C245A 2K X 8 ROMs. The ROMs are programmed to provide timing control within the FFT cycle for all of the functions of the FCC.

Some of the control signals out of the sequencer ROMs are dedicated lines like the 16-MHz clock out of pin 17 of ROM 16D. Other functions are encoded across several bits and decoded into individual timing signals by stages such as PALs 18D, 12D, or 13D. Some signals are timing signals required on the FCC itself, like the INIT1\ signal coming out of PAL 18D, while other signals go off the card and provide the FFT cards with timing control.

The NEW FRINGE signal also comes from the MCC and gives the FCC fringe cycle visibility. This fringe cycle visibility on the FCC is used to control some FFT card signals, such as FRLOAD, NEW FR, or FSCLOAD, and is used for a microprocessor interrupt where it allows the microprocessor to perform fringe cycle duties such as model computations.

2.1.2.2 The Trig Table Stages

There are 5 stages of trig table outputs from the FCC. Each trig coefficient is transmitted out to the FFT cards in a 550 numbering system (See Section 3.1.2.3), and hence trig tables require a 10-bit wide bus. The trig table generators are RAMs that the RTS can fill with pre-calculated sine and cosine values required by the FFT engine butterfly stages of the FFT card. Each stage in the FFT card has it's unique sine/cosine requirements. The RAM architecture provided by the FCC for each stage is appropriate for a given stage.

Early FFT butterfly stages require small tables repeated several times during the course of an FFT cycle while later stages have larger tables. In each case, RAM space for 128 complete tables per butterfly stage is provided. The 128 separate tables are cycled through (twice) in the course of the 256 FFT cycles of a 4.128 msec fringe cycle with the result that the FFT engine has the performance to be expected from trig tables with 7 additional bits of resolution.

The trig table storage provided for FFT butterfly stages 1, 2, and 3 are seen on sheet 2 of the FCC schematic. In each case, there is a counter that is synchronized by an FFT cycle sequencer INIT signal. Once initialized to the FFT cycle by an INIT signal, a counter free runs during the course of an FFT cycle.

Storage for the stage 1 trig tables is of a simple architecture since the small RAMs required are fast enough to run directly at the 32-MHz clock rate. Stage 2 and stage 3 have larger tables and require a more complicated memory with 2 banks of RAMs, each running at 16-MHz, tri-stated together to reach the system clock rate. The memory provided for each butterfly stage has enough RAM address space for the number of angles required by the stage (the address for this RAM space comes from the 74F163 counters) and a 7-bit address space for the 128 individual tables (actually, stage 2 and 3 have an 8-bit address space for 256 tables but only 128 are used in the system).

The individual tables are addressed by the binary counter in PAL 8A during the course of a fringe cycle. This counter increments every FFT cycle and hence starts each FFT cycle with a new trig table. The output of the 8A counter daisy chains through 8-bit registers at each stage (12A for stage 2, 17A for stage 3, 23A for stage 4, and 29A for stage 5). The resulting 8-bit wide, 4-bit long, shift register is sequentially clocked by the INIT timing signals from the FFT cycle sequencer so that each stage changes tables at the top of an FFT cycle (the start of an FFT cycle in a given trig stage is delayed by 11-bits from the stage just upstream from it to match the pipeline delay of the FFT card). The system cycles through the total of 128 tables twice every fringe cycle.

At each trig table stage there are 74ALS245 bus transceivers so that the FCC microprocessor can both write to and read contents of the trig table RAMs. The counters at each stage can be made transparent, as can the 8A counter, allowing the RAMs to be mapped into the microprocessor memory space. The microprocessor signals needed to write to or read from the trig table RAMs include (in some cases both polarities of the signals below are available),

1) PROG to make the 74F163 counters transparent (this signal also stops the FFT cycle sequencer stage INIT signals from PAL 18D).

2) LDTABLE to write to a RAM.

3) RDTABLE to read a RAM.

4) a RAMCS for the stage and RAM desired.

5) EVENRAM or ODDRAM to select stages with even and odd RAM banks.

6) TBMODE to make the 8A table counter transparent.

7) TABLE[0..7] to address a specific trig table (through 8A).

As with all 32-MHz control signals from the FCC card to the 10 FFT cards of a bin, the trig table signals all have 2 fanout versions. Each of these two versions drive five of the ten FFT cards in an FFT bin. The trig table output drivers are output enabled by signals A-ENA\ (for the A versions of the trig table outputs) and by B-ENA\ (for the B outputs). The two output enables are wired to the card connector and, in the system, are strapped to ground. In the FFT/MAC card test fixture, however, the A- and B- versions of the trig tables are tri-stated together and the test fixture microprocessor provides the A-ENA\ or B-ENA\ signals to select which supplies the drive for the trig table to the FFT card under test. In this way the FFT/MAC card test fixture can be used to better test an FCC. The 74F821 drivers (FCC 14A, 13A, etc.) that produce the A and B versions should be Fairchild/National instead of Signetics. Slight errors occasionally occurred with the Signetics chips.

2.1.2.3 The FCC Microprocessor

The FCC microprocessor can be seen on sheet 3 of the FCC schematic. The Dallas 87C520 has a number of functions on the FCC card,

1) communicate via an HCB port with the RTS.

2) load and verify the trig table RAMs.

3) load the FSTC Xilinx chip 3G with it's personality.

4) act as host for the hardware model generator.

5) load and verify the pulsar gate generator.

6) load the FFT card ASIC control words.

7) load the FFT card RAMs.

The HCB support is seen in the lower right hand corner of the FCC schematic, sheet 3. The HCB communication is interrupt driven via the 87C520 INT1 interrupt.

Note, the 2950 chip, used in the HCB section, was found to be unavailable. A pin compatible replacement was made by surface mounting a 29F52 and 74ALS74 on a small PC board. The schematic of the replacement is 1034d01.sch, stored in corrdwgs/hcb/SCCS. This replacement board is also used with the TRC and DEF.

2.1.2.4 The Hardware Model Generator

The hardware model generator is seen on sheet 4 of the FCC schematic. The logic consists of an 8K X 32 bit RAM parameter memory, an AMD29C327 floating point processor chip (fpc), shift register I/O to shift out 2-parameter fringe coefficients calculated by the model generator to the FFT cards, FIFO I/O for the fractional delay model parameters (and pulsar model parameters), and a micro-instruction sequencer.

The hardware model generator can be thought of as a special purpose microprocessor with a small, specialized instruction set. The 87C520 micro acts as host for the model generator, filling the parameter memory with double precision IEEE model parameters for the model computation and starting the model computation process by issuing the proper op-code to the model generator. Once started with an op-code, the model generators free-runs off the card 16-MHz clock to perform the instruction and halts upon completion.

The data sheets on the AMD29C327 floating point chip are contained in Appendix VI.

The fpc microcode is contained in /vlbsoft/pal_prom/proms/fccfpc. fccfpc.src in that area is the master source file, replacing the hardcopy master Orcad sheets that did not have electronic file masters. To print fccfpc.src from Unix, in landscape format, with page breaks at the correct places use: lwf -i0.5 -s7 -p -l fccfpc.src | lpr -Pprinter

To process fccfpc.src invoke make_fcc_micro_code. First, the processor (fccfpc.c) reads the fccfpc.src file. 8051 source files newrom0.src, newrom1.src, newrom2.src and newrom3.src are written. The .src files are processed resulting in .s19 files to be burned into the roms.

The processor for fccfpc.src reads both the binary and hex values and verifies that they are identical, in case manual editing is done to the hex field but the binary field is overlooked (or vice versa).

fccfpc.c requires the input file format to be precise. The only safe changes are in the precise binary and hex fields as positioned in the original version, and in the comments field on the right and at the bottom of each sheet.

See the readme.txt file in the area for additional details.

The principal instructions of the model generator are the fringe model computation, the delay model computation, and the pulsar model computation. All of these model computation instructions are basically the same with the main difference being in what happens to the end products. Each computes a 6-coefficient polynomial, with time as a variable, to obtain a phase result. Each then computes a 5-coefficient polynomial, in time, to obtain a corresponding rate result. The resulting phase and rate parameters are then used in other parts of the system to compute 2-coefficient models.

For fringe models, the end products are 2, 40-bit parameters that are shifted out to NCOs on the FFT cards. The shift registers are seen in the middle of sheet 4 of the FCC schematic (ICs 1D through 5D and ICs 1C through 5C). ICs 6H, 1H, 2H, and PAL 6C provide the interface required.

For delay models, the situation is a bit more complicated. The 16-bit integer part of the phase parameter is shifted out to the DEF cards via the 1C through 4C shift register string. ICs 6H, 3H, 4H, and PAL 6C and the TTL to ECL translator chips on sheet 1 of the schematic provide the interface. The 20-bit fractional part of the delay, and the 20-bit delay rate results are clocked into FIFOs 8H, 9H, 10H, 8F, and 9F to wait a new fringe cycle.

The pulsar model is like the fractional delay in that 20-bit phase and rate parameters from the model computation are shifted into FIFOs.

The micro instruction sequencer of the model generator is seen in the upper left hand corner of sheet 4 of the FCC schematic. The sequencer has a 512 X 64 bit micro-instruction memory that provides control signals for the various stuff that comprise the hardware model generator. A micro-instruction address counter is provided by PALs 33J and 33H. The PALs are started at the first micro-step of an instruction by the 87C520 microprocessor and then count through the rest of the micro-steps of the instruction. A capture register, 34J, and a small index counter in PAL 34H give the model generator limited looping capability.

Sheet 5 of the FCC schematic shows the AMD math chip (the fpc). The basic steps of a 6-parameter polynomial are seen in the lower left hand corner of the sheet.

The hardware model generator works as a slave computer for the 87C520

microprocessor and hence none of the activities undertaken by it are timed accurately enough for direct use by the correlator. All results of the model generator go into some type of secondary storage, on an FFT card in the case of the fringe model results, on a DEF card for the integer part of the delay, or in a FIFO on the FCC card itself. The results must wait in this secondary storage for a new fringe cycle to start.

Model computation is initiated by the 87C520 microprocessor writing a parameter memory start address in the parameter memory address counter 21J and 22J. This address in held in the counter until needed. The 87C520 also writes a destination for the model results in IC 6H (except for the pulsar model). This destination address will control which shift line is activated when the computed parameter shift-out is performed. Lastly, the 87C520 writes the op-code of the model computation instruction to the micro-instruction control PAL 34H and address counter PALs 33J and 33H. As the model computation progresses, the micro instruction sequencer controls the parameter memory counter to advance through the address of each model parameter as it is needed by the floating point math chip.

2.1.2.5 The FSTC Support Logic

When a delay model is computed by the model generator, two 20-bit coefficients are left in FIFOs (8H, 9H, 10H, 8F, and 9F) on the FCC. These coefficients are to be used in the computation of 2-parameter models to track the fractional sample error beginning at the start of the next fringe cycle.

The first of these 20-bit parameters is the fractional sample time correction (FSTC) to apply during the first FFT cycle of the new fringe cycle. The second 20-bit parameter is the rate of change of the first per 16-usec FFT cycle. In all, 20 such pairs of parameters for the fractional sample time correction, two for each FFT card in the FFT bin, are left in the FIFOs after the fringe cycle model computation. Having two pairs per fft card allows for two separate phase centers to be used. One additional pair of 20-bit parameters are left in the FIFOs. This pair is for the pulsar model which is handled in a manner similar to the FSTC model.

The parameters will be used during the course of the next fringe cycle to track a linear FSTC model just as the two 40-bit fringe model parameters shifted into the FFT cards are used by the FFT card fringe NCOs to track a linear fringe model. Whereas the FFT card fringe NCOs update the linear fringe model every clock cycle, the FSTC linear model is computed every FFT cycle.

The fractional sample time correction Xilinx chip on the FCC (see IC 3G on sheet 4 of the FCC schematic) is the 2-parameter model generator for the fractional delay correction factor. This chip does a linear extrapolation with the two 20-bit FSTC parameters every FFT cycle (16-usec). These extrapolations are accomplished by doing 20-bit serial additions in the FSTC chip, adding the rate parameter to the fractional delay parameter once an FFT cycle.

The FSTC Xilinx chip has 4 serial adders (see the FSTC Xilinx top level diagrams, in k027d0x.blk in Volume II). The operation of the FSTC Xilinx chip can be broken up into 4 basic operations:

1) eight 20-bit parameters are broadside shifted out of secondary storage FIFOs into the 4 serial adders of the chip (into the FSTC[0..19] bus).

2) the eight 20-bit registers in the Xilinx chip are made into shift registers and 20-clocks (actually 15 as per the note below) cause the parameters to circulate around the registers, LSB first. During this initial pass, FSTC initial phase parameters for the FFT engines are shifted into the FFT card NCOs (2 FFT cards at a time). During this initial circulation, the parameters are not changed, i.e. no serial addition takes place.

3) the eight 20-bit parameters are given a second set of 20-clocks and a second recirculation around the shift registers occurs. During this second pass, an FSTC slope parameter is shifted out to the FCC cards. Also during the second pass a serial addition occurs and the FSTC rate parameter is added to the delay parameter.

4) the eight 20-bit parameters, with the delay parameter now extrapolated for the next FFT cycle by the serial addition, are broadside shifted out of the FSTC chip back into the FIFOs (on the FST[0..19] bus).

In actual fact, steps 1 and 4, above, occur at the same time, i.e. as new parameters are shifted into the FSTC chip from the FIFOs, the parameters just used are simultaneously clocked back into the FIFOs. Also, in order to save clock cycles, there is a 5-bit offset wired into the broadside shift of step 1 and hence step 2, above, requires only 15-clocks to circulate the 20-bit parameters around the first time. The FFT NCOs get 5 clocks during step 1 and hence get a full 20-bit clock shift signal for the initial slope parameters.

At the start of a new 4.128 msec fringe cycle, parameters fresh out of the hardware generator are shifted into the Xilinx chip from the top row of FIFOs seen on sheet 4 of the FCC schematic (8H, 9H, 10H, 8F, and 9F). As the parameters are passed through the FSTC Xilinx chip, the serial additions are performed and the running sum results are stored into the lower row of FIFOs on sheet 4 (8G, 9G, 10G, 9E, and 10F). For the remaining 255 FFT cycles of the fringe cycle, the FSTC Xilinx chip works out of the lower FIFOs and the top row of FIFOs are available for secondary storage of newly computed FSTC model parameters waiting for the next fringe cycle.

To see how the FSTC stuff works, let us consider theory first. A constant time delay in a function being Fourier transformed causes a phase wrap to appear in the transform output (see "The Fourier Transform and Its Applications", second edition by R. N. Bracewell, page 104 and figure 6.6 on page 106). If the delay model were fully used in the TRC, i.e. both integer and fractional parts, no such time delay would exist and all transforms from all stations of the VLBA correlator could be directly cross multiplied together. The delay model can only be corrected for in the TRC/DEF cards to the 1-bit level, however. Since the samples have already been taken at processing time, nothing in the direct data path can be done to correct an error in sampling time of a fraction of a sample period. The result is that the Fourier transform spectrum obtained from one station will appear to have a phase wrap in it relative to a second station for which there is no fractional sample error.

The phase wrap in a station spectrum caused by a fractional sample period error can be removed by applying an opposite phase wrap to the output spectra at the final FFT butterfly stage. This correction would normally be done after the Fourier conversion to the frequency domain is complete but in the case of the VLBA correlator, the correction is done at the input to the final butterfly stage (for transforms of length where final radix-2 stages are not performed, this stage is in by-pass mode). In the final butterfly stage, the opposite phase wrap is applied to the trig twiddle factors with the result that the final transformed station spectrum appears to have come from a sampler phase shifted in time from the actual sampler by minus the fractional sample period error.

The overall operation of the fractional delay error correction can be gleamed from k027d02.blk in Volume II. This drawing shows the requirements for correction of an error caused by a fractional delay error. Consider first the lower part of the sheet, for small FFT sizes. For transform sizes of 64 point to 512 points, one FFT engine does the entire transform. In this case, what is required is that a phase ramp be applied to the output spectrum, starting at 0 for spectral point 0 and ending at the highest spectral point with the value of the 20-bit FSTC phase parameter (which is turned into a phase angle in the FFT

card FSTC look up table ROMs). Two 10-bit NCOs from the FFT engine are connected together for the 20-bit FSTC parameters. They are loaded with an initial phase of zero and a rate of the FSTC phase parameter divided by the number of spectral points to be obtained from the transform. The ramp shown in the lower left corner of k027d02.blk is the result. For spectral point N, a number, which the FFT card FSTC ROMs will interpret as an angle, and which is proportional to N, is applied to both the trig twiddle factors of the final butterfly yielding spectral point N. Through all points of the transform, this appears as a phase wrap opposite to the FSTC induced phase wrap error.

The requirement of the FCC FSTC Xilinx chip (3G) is to provide the FFT card NCOs with the 20-bit FSTC parameters. Returning to sheet 2 of the FSTC Xilinx design, the 2 serial adders, shown in block diagram form in the lower part of the sheet, program the 4 FFT engines of one FFT card. The 4 FFT engines on an FFT card perform transforms on the signals from two delay centers and hence two of the engines will get the output of each serial adder (this can be done since the two channels of one delay center have the same delay model). The parameters have already been divided by the number of spectral points to be obtained by the transform. All that is required is to shift a zero initial phase and a rate equal to the 20-bit fractional delay error (divided by the number of points expected from the transform) in the FSTC chip to the FFT card NCO.

For 1024-point transforms, things get a little more complicated. For each 1024 point transform, two FFT engines on an FFT card are required. Pipelines pair 0 and 1, and the pair 2 and 3, each do a 1K FFT. Pipelines 0 and 2 each do the even points, 0 through 510. Pipelines 1 and 3 each do the odd points, 1 through 511. At the final butterfly stage (which is in by-pass mode) the two engines see the odd and even spectral points, respectively. In order to do the FSTC stuff, the initial phase for FFT engine 0 (or 2) must get zero, but FFT engine 1 (or 3) will start at point 1 so the initial phase its NCO must get is 1 times the Xilinx chip rate parameter (the delay model fractional part divided by the transform size). See k027d02.blk in Volume II. Also, since each engine counts through the spectral points in two's, the slope for the FFT NCOs must be doubled. The 1-bit delay boxes in the center drawing of sheet 2 of the FSTC logic do this doubling (a 1-bit left shift).

For 2048-point transforms, things get a lot more complicated. Now four FFT engines on an FFT card are required. At the final butterfly stage of this hook-up, engine 0 outputs even spectral points 0 through 510, engine 1 outputs odd spectral points 1 through 511, engine 2 outputs even spectral points 512 through 1022 and, finally, engine 3 outputs odd spectral points 513 through 1023. In this mode, one of the 2 serial adders on the FCC is idle, and the other loads the four FFT engine NCOs with the following parameters (See k027d02.blk):

```
engine 0 phase = zero
engine 1 phase = zero
engine 2 phase = 512 times the fractional delay parameter
engine 3 phase = 512 times the fractional delay parameter

engine 0 rate  = 2 times the fractional delay parameter
engine 1 rate  = 2 times the fractional delay parameter
engine 2 rate  = 2 times the fractional delay parameter
engine 3 rate  = 2 times the fractional delay parameter
```

The NCOs were originally loaded with the phase parameters listed below. The 513 for engine 3 was found to not work for negative rates when they were introduced. The use of the existing 512 value was found to not cause any noticeable effects. Zero was used for engine 1 to be consistent with 512 being used for engine 3. The changes were implemented in pals 7E and 7F of the FCC.

```
engine 0 phase = zero
engine 1 phase = 1 times the fractional delay parameter
engine 2 phase = 512 times the fractional delay parameter
engine 3 phase = 513 times the fractional delay parameter
```

The phase parameter loaded in a given FFT engine FSTC NCO depends on the first point that engine will output during an FFT cycle (point 0 for engine 0, point 1 for engine 1, point 512 for engine 2, and point 513 for engine 3) but all engines count through their set of output spectral points in two's and hence all FFT engine NCO rate parameters are 2 times the fractional parameter. Remember that the delay parameters referred to above were divided by 1024 before being stored in the FIFOs by the hardware model generator.

The factor of 512, above, is obtained in the 9-bit shift register seen in the top part of k027d02.blk as fstc6 (a 9-bit left shift). The 513 factor was to be obtained by doing a serial add of the delay parameter (fstc0), and the 9-bit shifted version of the delay parameter (fstc6), to obtain fstc7. However the addition is done incorrectly for negative numbers, so fstc6 is used, in the PALs, as an approximation of fstc7. (See Section 2.2, Overview of FCC Model Implementation, below.)

Sheet 3 of the FSTC logic diagram (k027d03.blk in Volume II) summarizes the stuff described above and gives a timing diagram to show how the actual NCO parameters are shifted out into the FFT card. The shift out of parameters to the FFT cards is complicated by the FFT card NCO's requirement to shift out 10-bits of phase, followed by 10-bits of rate, followed by the remaining 10- bits of phase, followed by the remaining 10-bits of rate. Sheet 3 of the FSTC logic breaks things down twice by FFT engine and FFT size. Good luck in following this sheet.

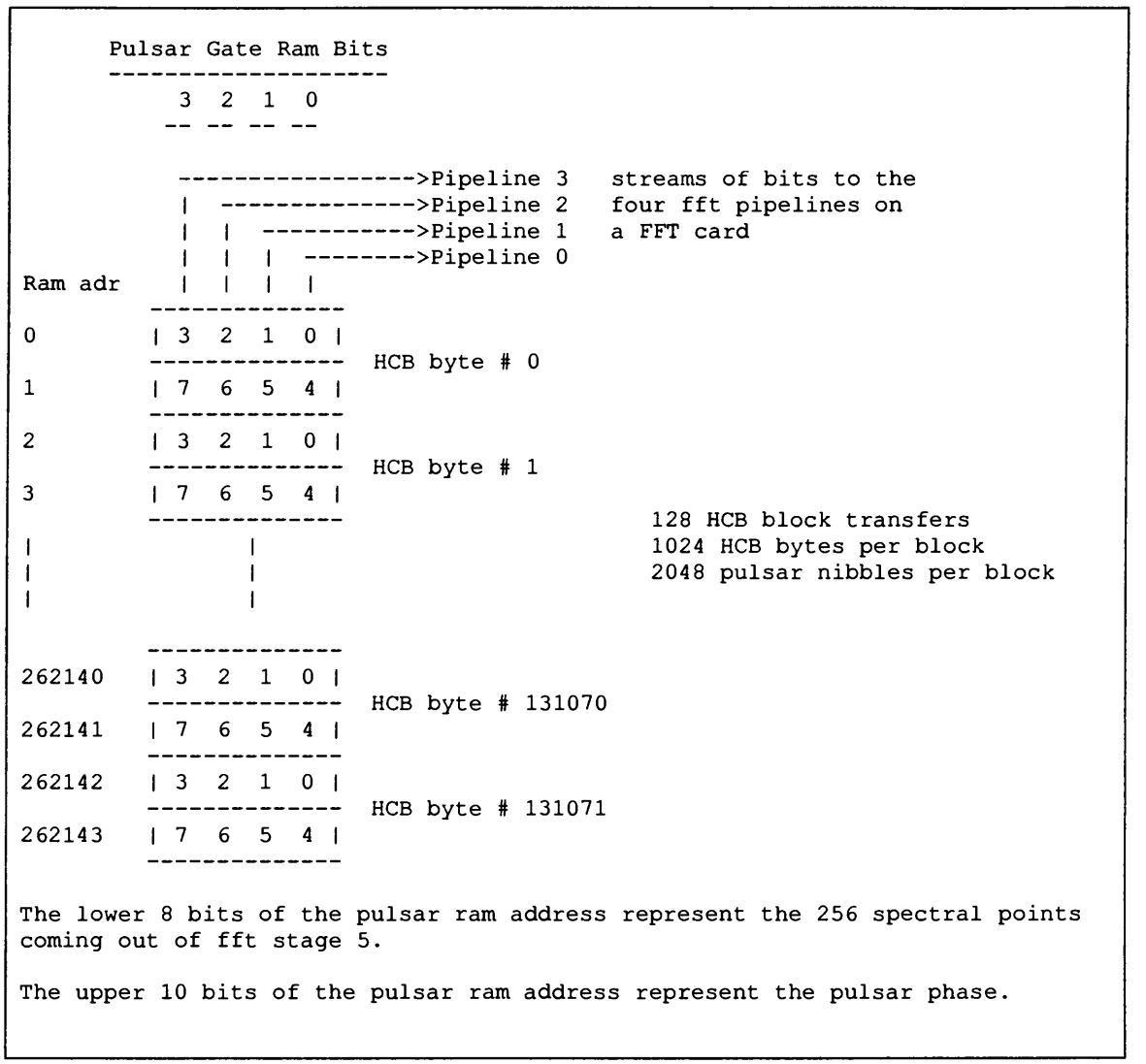
PALs 7F and 7E on sheet 4 of the FCC schematic assist in the work of patching the various 10-bit segments together for the FFT card NCOs. The RAM 1E supplies these PALs with an FFT size code and the FFT cycle sequencer provides the timing signals. The 74ALS138 8E provides the shift clock to shift the FSTC parameters into the FFT card NCOs. The shift clock is controlled by timing signals from the FFT cycle sequencer. FFT cards are loaded 2 at a time (from the 4 serial adders on the FSTC chip) so they can get through all of the parameters required in a single FFT cycle.

The 2-parameter pulsar model is handled in the same way as the fractional sample correction except that the resulting pulsar phase is shifted into IC 1B for the pulsar gate generator.

2.1.2.6 The Pulsar Gate Generator

The pulsar gate generator is seen in the upper right corner of the FCC schematic L013D04.SCH. It provides the FFT engines with discrete signals that reflect the status of a pulsar (i.e. is the pulse on or off) for each spectral point out of the FFT engine during pulsar observations.

The pulsar gate mask lookup table is contained in a 256K X 4 RAM, distributed among four 64K X 4 RAM chips. See the following ASCII diagram. An 8-bit counter (ICs 3B and 4B) counts through the 256 spectral points of an FFT cycle, after being synchronized to the FFT cycle via the FFT cycle sequencer INIT signal. The 10-bit register 2B holds, for a given FFT cycle, the pulsar phase. Thus, for a given spectral point and a given time (where the 2B IC specifies the pulsar phase for that time), the look-up bit stored in the RAM gives the on/off status of the pulsar. Logic 0 enables ("ON"). Logic 1 disables ("OFF"). The secondary storage IC 1B gets a serial pulsar model from the model generator every FFT cycle (16-usec) via the FSTC Xilinx chip.



Thus the pulsar gate generator has 256 values of the pulsar state, one for each spectral point, for each specific phase of the pulsar. The pulsar phase is quantized to the 10-bit level. The 4-bit wide pulsar gate output goes to the 4 FFT engines on an FFT card (and to all FFT cards in the FFT bin). The ASIC control words, for each FFT card, specify whether to enable the pulsar gate control, for that card. If 1024-point or 2048-point transforms are being performed, the spectra output by an FFT card is split between 2 or 4 FFT pipelines. The 4-bit wide pulsar gate RAM output provides pulsar gates across the entire output spectrum.

The outputs of the pulsar gate generator are from ICs 6A and 7A. The 6A output presents -A and -B versions of the gate signal that each drive 5 of the 10 FFT cards in an FFT bin. The 7A chip outputs are pulsar gate signals for the pulsar gate validity counters on the rack MCC card.

IC 6B is the microprocessor interface into the pulsar gate generator. By using this interface, the microprocessor can write to or read the contents of the RAMs. When the microprocessor controls the RAMs, the 10-bit page address is supplied by IC 5B.

2.1.3 FCC Microprocessor Software

The 87C520 microprocessor on the FCC card uses a separate 32K X 8 RAM chip as its main memory. The main program for the FCC microprocessor is downloaded by the RTS into this memory. There is a 16K EPROM as part of the microprocessor. The first 4K are used for permanently resident routines. The original 87C51 only had 4K of EPROM, which is the reason the rest of the 16K has not yet been used.

2.1.3.1 Brief Descriptions of the Software Modules

Below is a brief description of each of the modules listed in the chart below. (Memory allocations seen below are as of 8/26/97)

L O A D M A P			
Section Name	Starting Address	Ending Address	Size
master.obj	0 to FFF in ROM		
master_section	0000	0001	0002
int0_section	0003	0005	0003
timer0_section	000B	0010	0006
int1_section	0013	0015	0003
timer1_section	001B	0020	0006
uart_section	0023	00E6	00C4
monitor.obj			
monitor_section	00E7	0764	067E
romhcb.obj			
romhcb_section	0800	090C	010D
fft.obj			
fft_section	090D	09F5	00E9
test.obj			
test_section	09F6	0E52	045D
obs.obj	1000 and up in RAM		
obs_section	1000	12FA	02FB
ramhcb.obj			
ramhcb_section	2000	2A3D	0A3E
ram.obj			
ram_section	3000	3024	0025
ramtest.obj			
ramtest_section	3025	3209	01E5
help.obj			
help_section	4000	4CB9	0CBA

2.1.3.1.1 MASTER.ASM

MASTER.ASM has the software executed after a hardware reset to the microprocessor. This software will initialize the card and initialize microprocessor functions like the interrupt logic, the serial port, etc. This module also has the interrupt vectors. After a reset, the micro will execute from an idle loop in MASTER.ASM awaiting instructions from the terminal or RTS conveyed by interrupts.

2.1.3.1.2 FFT.ASM

This module has subroutines to load the FCC parameter memory, to load the AMD math chip mode instruction, and to load the FFT card ASIC control words. A table of idle ASIC control words, for use after power-up, is contained in the module.

2.1.3.1.3 MONITOR.ASM

The monitor software package has various terminal options supported by the FCC microprocessor, such as display and modify memory, etc.

2.1.3.1.4 ROMHCB.ASM and RAMHCB.ASM

ROMHCB.ASM and RAMHCB.ASM handle the microprocessor communications with the RTS, over the HCB. Some of the basic functions, such as memory load, are handled in the ROM based software, but most protocol support is executed out of the RAM based RAMHCB.ASM program.

2.1.3.1.5 TEST.ASM and RAMTEST.ASM

The TEST.ASM module has a number of hardware test functions that allow an operator to test the FCC card either in the FFT test fixture or in the system. Hooks are in place so that software tests can be written in RAM in the RAMTEST.ASM module. This capability allows the generation of new tests without changing the FCC ROMs.

2.1.3.1.6 OBS.ASM

OBS.ASM is the main observing software for the FCC microprocessor. This module is entered upon reception of an OBSERVE command from the RTS. The first action of the software is to configure the FCC card for the observation.

Once the card is configured, the FCC microprocessor starts the observation by enabling interrupts. All of the actions required to support the observation are executed out of interrupt handlers. The interrupt system is described in more detail below. When not supporting an observation in an interrupt routine, the microprocessor waits in an idle loop in OBS.ASM.

2.1.3.1.7 RAM.ASM

RAM.ASM is mainly used for linking the ROM and RAM based programs of the FCC together. It is not desirable to have to change ROMs on the FCC cards because on a change in a RAM based program. Thus direct linking of the ROM and RAM based modules is not possible. The strategy used in the FCC software is to always link from the ROM software into the RAM software via jumps to locations that are on page boundaries in the RAM modules. These jumps will not change memory locations as software changes are made to the main bodies of the RAM modules and hence link address in the ROM modules will not change because of a change to a RAM subroutine.

2.1.3.1.8 HELP.ASM

This module contains ASCII terminal help screens. MONITOR.ASM has an option to display the help screen. Four help screens are provided.

2.1.3.2 Microprocessor Interrupt Structure

Almost all of the functions provided by the FCC software are accomplished in interrupt routines. There are several interrupts and a summary is given in the following table.

interrupt	function	priority
SERIAL	terminal	low
INT0	fringe cycle	high
INT1	HCB communication	medium
TIMER0	not used	-
TIMER1	not used	-

The serial port interrupt is not normally used during an observation but supports a terminal and provides the functionality of the MONITOR.ASM software.

The INT0 hardware interrupt is the 4.128 msec fringe cycle interrupt. Each time this interrupt is received, the microprocessor uses the hardware model generator to compute new fringe, FSTC, pulsar, and delay models for the next fringe cycle.

The fringe models are run starting at label FRINGE in OBS.ASM. FRINGE consists of looping through the models for all 10-stations in the FFT bin. Each station has a set of model coefficients resident in the parameter memory that represent the active fringe model for the current fringe cycle (there is storage in the parameter memory for four complete sets of fringe models and a table in RAM, starting at memory location 7D00, indexed by the station number, gives which of the 4 is the current active model as per an assignment made by the RTS via the RAMHCB.ASM subroutine). An inner loop in the FRINGE subroutine loops through the 4 channels of the 10 stations. As the fringe models are run, 40-bit fringe phase and fringe rate parameters are computed and shifted into secondary storage on the FFT card.

At the label FSTC, after the fringe models are run, the Fractional Sample Time Correction section is run. The routine loops thru all 10 stations sending out fractional delay model parameters. Each station has two phase centers. By doing fractional delay stuff now, the FSTC parameters have the same time as the fringe models. Later, delay models are run again with t = t + 1 for the integer delay. In this way the signal will meet the proper models at every point in the correlator as it pipelines its way through the system.

Next, at the label PULSAR, the pulsar model is computed (even if a pulsar observation is not being run).

At the label TIME, before the delay models are run, the time parameters used in the computation of model polynomials is changed. The changes in the time parameter between the fringe and delay model computations reflects the pipelining in the VLBA correlator. By the time stuff gets out of the DEF card with a delay model computed with a given time, a fringe model in the FFT card computed with the same time will act on it.

In parameter memory (See Appendix III), times are stored for each of ten stations, plus a pulsar model, for each of the four models. Only one of the four models is the active model. Other models can be updated ahead of time for future changes of the active model. The routine INCTIME, called in the TIME section, increments the times for the ten stations and pulsar of the active model.

At the label DELAY, the delay models are run by looping through 10-stations for 2-delay centers of an FFT bin. Because of the mux at the input of an FFT card, the destination of the delay model output parameters must be programmable. There is a table at the end of the RAM.ASM module (label PBD) that gives the delay parameter destinations. With this programmability, the DEF card that drives FFT card x gets the delay model for station x.

Each delay model computation results in a 16-bit integer delay value being sent to the appropriate DEF (programmed by IC 6E on the FCC schematic L013D04.SCH) and two 20-bit FSTC parameters being loaded into the FIFOs on L013D04.SCH. Again, there is space for four complete sets of delay model parameters in the parameter memory and the table starting at memory location 7D00 gives the specific model assignment made by the RTS (See Appendix III).

When self test is to be run, special fringe and delay models are computed and the results are sent to the FFT and DEF cards. Only the 0th term of these self test models can be non-zero. This forces the model to not be a function of time during self test. Self Test (testing while observing) has not been implemented yet.

The INT1 hardware interrupt is used for RTS HCB communications with software support provided by the ROMHCB.ASM and RAMHCB.ASM software.

2.1.4 FSTC Xilinx Design

The FSTC Xilinx design used on the FCC is described in the logic diagram k027dxx.blk in /home/azalea/corrdwgs/fcc/sch/SCCS. k027d02.blk and k027d03.blk are included in Volume II. The discussions below will describe these schematic sheets. The diagrams were made using "home made" logic symbols but the schematics should still be followable with some effort. Each Xilinx CLB (configurable logic block) is shown in the logic diagrams as a box with two flip-flops just to the right (along with the various programmable multiplexers in the CLB). The box represents a Xilinx PLA with 5 external inputs, two feedback terms and two outputs. The heavy (bus) lines inside the boxes indicate which logic terms are included in the equation for a given PLA output. The specific equation can be obtained from the .MAC file used for the design or the Xilinx .LCA file.

Sheet 1 of the schematic shows the floor plan layout of the chip. The design was made with an XC-3030 (10-CLB by 10-CLB) chip. Sheets 2 and 3 of the schematic have already been described (see section 2.1.2.5).

At the bottom of sheet 5 of the FSTC schematic, a block diagram of the operation of the chip for one FFT engine is seen. In this block diagram, the parameters are seen being broadside loaded into the FSTC chip from the FIFOs at the bottom (or being broadside unloaded back into the FIFOs at the top). Once the 20-bit registers are loaded, the direction of shift changes from vertical to horizontal and the 20-bit parameters are recirculated around the horizontal registers twice. On the second recirculation, the serial adder seen on the left is active and the top (phase) parameter is updated by having the bottom (rate) parameter added to it.

There is a 5-bit wired offset between in the LOAD PARAMETERS input from the FIFOs and the 20-bit shift registers that is not shown on the sheet 5 block diagram. This wired shift saves time since only 35 clocks will be required to perform the two complete recirculations of the 20-bit parameters described above.

The 5-bit offset comes from the minimum FFT size of 64 points. The 20-bit phase parameters are divided by the number of spectral points to be obtained by the transform, before being loaded into the FSTC Xilinx chip. This dictates there will be a minimum of five bits leading zeros in the phase parameters. The phase parameters are positive, since the fractional part of the delay is always positive. See 2.2.3 for a more detailed explanation of delay signs. Since the phase is positive, sign extension is accomplished by leading zeros. The rate parameters have sign extension set by the maximum delay change rate. The rate parameters can be positive or negative.

The top 20-bit shift register has an output mid way down it from which

a 9-bit shifted version of the parameter can be obtained. This 9-bit shifted version represents the initial phase times 512 term required by one FFT engine for 2048-point transforms. Sheet 5 of the FSTC schematic also shows the 4 serial adders.

Sheets 6 and 7 of the FSTC schematic show the output stages of the FSTC chip. Sheet 6 also shows in block diagram form how the 10-bit chunks are shifted out of the chip on the 40 clocks required to shift both the phase and rate parameters into the FFT cards. The parameters go out LSB first and the 512 times parameter has nine leading (least significant) zeros shifted out first.

The action of sheets 6 and 7 are complicated by having to know where the various segments of the phase and rate parameters (sometimes multiplied by 1, sometimes by 2, sometimes by 512, and sometimes needing another serial adder ,CLB CJ on sheet 6, for a 513 times term) are when the time comes for a given segment to be shifted out. Sheet 4 of the FSTC schematic was drawn to help the selection process. Note the 513 term is currently ignored by the 7E/7F pals.

Sheets 8, 9, and 10 show four of the 20-bit shift registers. For example, CLBs AA, AB, AC (from sheet 8), CLBs AD, AE, AF, AG (from sheet 9) and CLBs AH, AI, AJ (from sheet 10) are connected in series to yield a 20-bit shift register for a phase parameter. Sheets 11, 12, and 13 show the other four 20-bit registers.

The design files for the FSTC Xilinx are maintained in the vlbsoft/xilinx/fstc/SCCS directory. The top level macro file, fstc.mac, contains the instructions for Xilinx generation of the fstc.mcs hex file that is downloaded. XACT version 4.3, and APR version 3.3 were used for the present design. The README.fstc file lists the files that are under SCCS version control.

2.2 Overview of FCC Model Implementation

2.2.1 Introduction

This section describes the model tracking methods used by the VLBA correlator hardware down to the bit level. Model change rates given are for earth based stations. The following figures apply: Page

2.2.2 Fringe Phase Model

Fringe model computation in the VLBA correlator is performed on three levels. The top level model generation is performed in the VLBA correlator VME computer system and results in coefficients to be used by a hardware floating point processor located on the FFT control card. This is updated every several minutes.

The hardware floating point processor implements the second level of fringe modeling by periodically evaluating two polynomial expansions for each channel being processed in the correlator. These two expansions result in the production of two 40-bit fixed point parameters that are then loaded into a 40-bit NCO associated with each FFT engine. This is updated every 4 ms.

The FFT card based NCO (Number Controlled Oscillator) is the third and last level of fringe model generation in the VLBA. The two 40-bit numbers produced by the FFT control card represent the fringe phase and fringe rate that are to be used by an NCO to generate sample rate fringe phase estimates via

implementation of a two coefficient polynomial. This updates at the 32 MHz clock rate.

The form of the two polynomials that are evaluated by the FFT control card can be seen below:

```
P = P0  +  P1T  +  P2T**2  +  P3T**3  +  P4T**4  +  P5T**5
```

where:

```
P0      DOUBLE PRECISION (IEEE FLOATING POINT)
P1      DOUBLE PRECISION
P2      DOUBLE PRECISION
P3      DOUBLE PRECISION
P4      DOUBLE PRECISION
P5      DOUBLE PRECISION
-------------------------
12 32-BIT WORDS PER MODEL
```

and

```
R = R0  +  R1T  +  R2T**2  +  R3T**3  +  R4T**4
```

Note R is the derivative of P. T is in 4 ms cycles.

where:

```
R0      DOUBLE PRECISION
R1      DOUBLE PRECISION
R2      DOUBLE PRECISION
R3      DOUBLE PRECISION
R4      DOUBLE PRECISION
-------------------------
10 32-BIT WORDS PER MODEL
```

The result of these two polynomial equations, after the conversion to 64-bit fixed point, is two quantities as seen below representing the fringe phase and its rate of change per sampling period (note decimal points).

```
Fringe Phase                    /--Decimal Point
    FFFFFFFF,FFFFFFFF,FFFFFFFF.FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF

Fringe Rate of change  (Decimal Point at far left)
   .SSSSSSSS,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR
```

RRRR is sign extended when a negative number. The largest negative number would be .SRRRRR.

The parameters P0-5 and R0-5 can be set via the RTS functions setfpc and setfrc. The RTS function setfpc includes a multiplication of the coefficients by 2**40 to correctly position the decimal point. Similarly, setfrc multiplies the coefficients by 2**64 for positioning the rate decimal point.

```
setfpc(card,modelnr,channr,coefnr,value) /* scaled by 2^40 */
setfrc(card,modelnr,channr,coefnr,value) /* scaled by 2^64 */
```

Parameters for the two coefficient model generator are obtained from these two results by doing a 24-bit left shift on the top number to discard the integer portion of the phase. The remaining values are then truncated to 40-bits and sent to the FFT card NCOs. The form of the two coefficient computation at the FFT card is seen below (where FFF---FFF are the 40-bits of fringe phase, SSS---SSS are extended sign bits of the fringe rate and RRR---RRR are the significant bits of the rate);

```
  .FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF  FRINGE PHASE (IN TURNS)
+ .SSSSSSSS,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR  FRINGE RATE (IN TURNS/
  ------------------------------------ --------           SAMPLE PERIOD)
  .PPPPPPPP,PPPPPPPP,PPPPPPPP,PPPPPPPP,PPPPPPPP  UPDATED FRINGE PHASE
   ^--------^                                    9-BIT ACTIVE RESULT
```

where, for example,

```
  .00000000,00000000,00000000,00000000,00000000  IS ZERO TURNS
  .01111111,11111111,11111111,11111111,11111111  IS (.5 - ONE LSB) TURNS
  .10000000,00000000,00000000,00000000,00000000  IS 0.5 TURNS
  .11111111,11111111,11111111,11111111,11111111  IS (1 - ONE LSB) TURNS
```

The 9-bit active result indicated above represents the 9 most significant bits of the fringe phase NCO result that is actually used by a look up table ROM, on the FFT Card, to do complex multiplication of the incoming samples.

The number .SSSSSSS,RRRR... is added to the nco each clock cycle. For example, let R0, the zero term of the rate polynomial, equal 001. The msb represents a positive sign. There is an implied decimal point at 0.01. This gives a value of R0=+.25. So adding this value of R0 to the nco each 32 MHz clock gives an 8 MHz nco msb frequency. This could be directly measured with an oscilloscope.

Having a negative fringe phase is permissible as shown via the example below. We show that we can subtract an integer, getting a negative result. The fractional part ends up unchanged.

case 1)

station	total fringe phase (in turns)	fractional part (2's complement)
antenna 1	123.456	.0111010010111100
antenna 2	1000.123	.0001111101111100
antenna 3	5678.876	.1110000001000001

case 2) (same relative fringe phases but with a different reference)
(There are -654 turns added to each value)

station	total fringe phase (in turns)	fractional part(2's complement)
antenna 1	-530.544	(see below)
antenna 2	346.123	.0001111101111100
antenna 3	5024.876	.1110000001000001

To finish the chart above, first convert the -530.544 into hex, and then into binary and then into 2'complement binary.

```
-530.544 decimal = -212.8B439 hex = -0010 0001 0010.1000 1011 0100 0011 1001
                 one's complement = 11101 1110 1101.0111 0100 1011 1100 0110
                                  +            1.0
                                    ---------------------------------------
                 two's complement = 11101 1110 1101.0111 0100 1011 1100 0110
```

station	total fringe phase (in turns)	fractional part (2's complement)
antenna 1	-530.544	.0111010010111100
antenna 2	346.123	.0001111101111100
antenna 3	5024.876	.1110000001000001

which is the same as case 1.

See Section 2.2.5 for magnitude scaling considerations.

2.2.3 Delay Model

As with fringe model computation, delay model computation in the VLBA correlator is performed on 3 levels with the top level computation being done in the VME computer system.

The next level of delay model computation is done on the FFT control card in the floating point model generator. Delay models at this level are computed every 4-msec and require the evaluation of two polynomial equations for each pair of channels. The last level of (fractional bit) delay modeling is done on the FFT control card and employs a hardware implementation of a two coefficient model generator analogous to the FFT card NCO. See Section 3.2, The FFT Card Description, for a detailed description of the FSTC (Fractional Sample Time Correction).

The form of the first polynomial that is evaluated by the FFT control card can be seen below:

```
D = D0  +  D1T  +  D2T**2  +  D3T**3  +  D4T**4  +  D5T**5
```

WHERE:

```
D0      DOUBLE PRECISION (IEEE FLOATING POINT)
D1      DOUBLE PRECISION
D2      DOUBLE PRECISION
D3      DOUBLE PRECISION
D4      DOUBLE PRECISION
D5      DOUBLE PRECISION
M0      DOUBLE PRECISION (MODE DEPENDENT SHIFT PARAMETER
-------------------------        DESCRIBED LATER)
14 32-BIT WORDS PER MODEL
```

The evaluation of this polynomial results in a value for the delay of a channel which when converted to a 64-bit fixed point number is of the form;

```
00000000,00000000,IIIIIIII,IIIIIIII.FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF
```

where;

```
00000000,00000000,IIIIIIII,IIIIIIII. IS THE INTEGER PORTION OF THE DELAY
```

and

```
.FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF  IS THE FRACTIONAL PART OF THE DELAY
^--------------------^                20-BIT RETAINED RESULT
```

Note the fractional part of the delay is always positive.

The 16 bit integer part of the delay can be positive or two's complement negative. OVLBI requires the integer part having the ability to be positive or negative.

Following are some examples of the way the FCC represents numbers:

Number to be represented (+ or -)	=	Integer Part (+ or -)	+	Fractional Part (always +)
5.3	=	5	+	0.3
-5.3	=	-6	+	0.7
5.0	=	5	+	0.0
-5.0	=	-5	+	0.0

The 16-bit integer part of the delay is sent to and applied at the playback interface logic. The integer delay is limited to 16-bits at this point in the correlator because the serial data path between the model generator and the deformater card can only handle 16-bit numbers. Larger accumulative delays must be handled via another VME-PDB link. The most significant 20 bits of the fractional sample part of the delay are kept and used in the two coefficient polynomial model generator to produce fractional sample error corrections to be applied at the FFT engine output.

The two coefficient model requires the computation of a second polynomial to yield the rate of change for the fractional part of the delay to a resolution of 20-bits. The form of the second polynomial is seen below;

R = R0 + R1T + R2T**2 + R3T**3 + R4T**4

where:

R0	DOUBLE PRECISION
R1	DOUBLE PRECISION
R2	DOUBLE PRECISION
R3	DOUBLE PRECISION
R4	DOUBLE PRECISION

10 32-BIT WORDS PER MODEL

the result of this equation is converted to fixed point and the 20 MSbits retained.

The two 20-bit numbers (the fractional sample error described earlier and R, the rate of change of this number described above), are then used by logic on the FFT control card to generate a 20-bit fractional sample ramp slope every 16-usec. This ramp slope is then loaded into 20-bit NCOs on the FFT card.

The 20-bit fractional delay is computed as a two's complement number between plus 1 and minus 1. The format is S.xxxxxxxx,xxxxxxxx,xxx.

The math in the Xilinx has been changed so it can handle negative numbers, due to OVLBI. A negative rate would give a negative number, which is ok. There used to be a half bit added. It was eliminated by the following change in fft.asm change the statement: FDB 4000H ;DELAY MODEL 0.5 #3
to: FDB 0000H ;DELAY MODEL 0.5 #3

As an example of nco operation, let Phase=fractional part of the delay polynomial evaluation =0.5 bits. If looking at ANGLE0 out of the nco, where ANGLE7 is the msb, Phase=0.5 bits gives 2 MHz out on the ANGLE0 line. That is obtained by adding 0.00000000,1 to the nco 256 times in 16 usec to obtain 0.10000000,0 (or 0.5 bit).

Now, let's simulate Phase, or fractional part of the delay = 1.50 bits. This is desirable for OVLBI applications. The only way to get phases >= 1 is by the additions of R for subsequent fft cycles in the 4 ms. cycle. This causes an nco rate that has overflows in the msb. That is obtained by adding 0.00000001,1 to the nco 256 times in 16 usec to obtain 1.10000000,0. The msb overflows and is lost. ANGLE0 gives 6 MHz if looked at with an oscilloscope. If R is negative, delays of less than zero can result.

Larger values of R can apply delay adjustments of greater than one bit. There is one transition in the nco msb, per bit delay. A transition is 0 to 1, or 1 to 0.

The following table shows the correction applied as a function of NCO bit.

nco bits 7654 3210	delay (bits)	phase rotation @ high band edge (degrees)	twiddle correction (degrees)
0000 0000	0.0	0	0
0100 0000	+0.5	+90	-90
1000 0000	+1.0	+180	-180
1100 0000	+1.5	+270	-270
1111 1111	+2.0-	+360-	-360+

This table indicates that as the nco MS 8 bits count from 0 to 255, the associated delay range is a total of 2 bits. Thus, if during a single FFT cycle, we see the bits start at all zero's and count to all one's and overflow back to zero, this means the correction applied in that cycle is something greater than two bits. A correction of almost one bit would require that the nco count from 0 to near 127. Note that since +360 degrees is the same phase as 0 degrees, it is ok for the nco to overflow.

For example, consider if the initial phase is 0.5 bit. A rate value R = .5/256 gives no transitions of the msb. (It is added 256 times, since there are 256 fft cycles per 4ms fringe cycle) R=1.5/256 will cause a transition in the msb half way through the fft cycle at the end of the 4ms cycle. R=-2.5/256 gives similar results.

Another Example:
20 bit FSTC NCO on the FFT card:

128	64	32	16	8	4	2	1												
7	6	5	4	3	2	1	0												
19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1											
0	0	0	0	0	0	0	1	0											
0	0	0	0	0	0	1	0	0											
1	0	0	0	0	0	0	0	0											

note A: note A rate 1

note B: note B rate 2

note C: note C rate 3

result 1 = 256 * rate 1

note A:
If this value is added to itself 256 times, the result at the end of the fft cycle is 1000 0000 0. Referring to the third row of the above Table, the correction applied to spectral channel #255 is -180 degrees, half a turn, for 1 bit.

note B:
If this value is added to itself 256 times, the result goes to all zero's so the correction is -360 degrees, one turn, for 2 bits.

note C:
This one should cause the nco to ramp up once, return to zero and ramp up a second time, two turns, for 4 bits.

To enter the initial phases (D0-5) or rates (R0-4) use the following

RTS functions. The function dpc implements the initial phase scaled directly in bits. The function drc gives the rate added by the FSTC Xilinx to the initial phase.

```
dpc(ant,modelnr,dlynr,coefnr,value)     /* scaled by 2^32 */
 drc(ant,modelnr,dlynr,coefnr,value)     /* scaled by 2^63 */
```

The pre-OVLBI correlator specification states a maximum of 50 bits of delay change per second. Thus, the delay can change by no more than 0.00080 samples in one 16-usec FFT cycle, or by 0.2048 samples in one 4-msec fringe update cycle. The playback speed-up factor need not be considered here. That is since 32-Msamp/s, which produces the highest rate of delay change, must be recorded at the highest record rate, and hence will have a unity speedup factor.

The precision used in the two coefficient FSTC model generation are seen below (again SSS---SSS are extended sign bits);

```
  F.FFFFFFFF,FFFFFFFF,FFF   FRACTIONAL DELAY CALCULATED BY MODEL
+ S.SSSSSSSS,SSRRRRRR,RRR   DELAY RATE
```

The fractional delay parameter represents the rotation angle to be applied to the highest number spectral point of the output spectrum. The number that is loaded into the FFT card ramp generator is proportional to this number but is dependent on the transform size being performed since it must be added to itself N/2 times for an N-point transform. If a 128-point transform is being performed, for example, the number shifted out to the FFT card NCO is;

```
0.00000FFF,FFFFFFFF,FFF   128-POINT TRANSFORM SLOPE
```

where the corrections being applied at the transform output will be;

```
0.00000000,00000000,000   ROTATION ANGLE APPLIED TO SPECTRAL PT 0
0.00000FFF,FFFFFFFF,FFF   ROTATION ANGLE APPLIED TO SPECTRAL PT 1
0.0000FFFF,FFFFFFFF,FF0   ROTATION ANGLE APPLIED TO SPECTRAL PT 2


(F.FFFFFFFF,FFFFF000,000   ROTATION ANGLE APPLIED TO SPECTRAL PT 63*
-0.00000FFF,FFFFFFFF,FFF)
 ^--------^                8-BITS USED BY FSTC ROM ( +1 TO -1 Bit since
                           it can go negative)
```

* Note this value is (64 - 1) times the spectral point 1 value.

For OVLBI, a change was made to the Xilinx personality to allow negative rates. Thus, a sign bit needs to be allowed in the initial phase, for the two's complement arithmetic. This can be thought of replacing the initial F with a S.

The model parameters active in the two coefficient model generator therefore becomes:

64-POINT FFT

```
S.FFFFFFFF,FFFFFFFF,FFF   Xilinx initial phase
S.SSSSSSSS,SSRRRRRR,RRR   Xilinx Rate
```

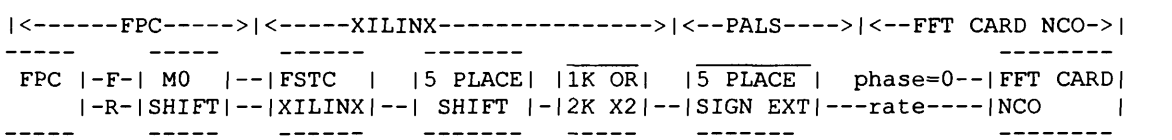
Note, since we now are allowing the sum to go negative, we will get an error if the msb of the initial fractional delay is 1, since that is our sign bit. This is what broke the 64 point fft. By adding .5 bit, we sometimes had the msb=1, which we need for the sign bit. Deleting the adding of .5 bits, fixed the problem. Then, the initial fractional delay will always be less than one, and the msb will always be 0, defining a positive number. The fractional part of initial phases are always positive, as defined earlier in this section.

A mode dependent shift is specified by M0, the Mode Dependent Shift Value from Parameter Memory. M0 is defined:

M0	FFTSIZE	FSTC NCO RATE	# FFT PIPELINES USED	# ADDITIONS PER FFT CYCLE (16uSEC)	# ADJACENT TRANSFORMS
0	64	2 MHz	1	32	8
1	128	4 MHz	1	64	4
2	256	8 MHz	1	128	2
3	512	16 MHz	1	256	1
4	1024	16 MHz	2	512	1
5	2048	16 MHz	4	1024	1

The larger the fft size, the more times the rate shifted into the nco is added to the accumulator in an fft cycle. The regulation of the number of additions is controlled by regulating the nco frequency as a function of fft size. This is accomplished by a clock enable to the fft card nco. The spectral results out of a pipeline for an fft size < 512 are grouped so that successive fft groups use the same fstc nco output, before the nco steps to the next value. Spectral points 0 from all adjacent small transforms grouped within the 256 points are grouped together. Then all the spectral points 1 are grouped together. etc. For 1024 and 2048 point ffts, the extra additions are obtained by using more pipelines.

The drawing below shows where the various manipulations take place. FPC stands for the AMD Floating Point Chip.



The M0 shift must be performed on both 20-bit coefficients, by the floating point processor, before they are passed to the two parameter FSTC Xilinx.

Also, a shift of 5 places is performed on the slope parameter at the Xilinx two parameter model generator shift out into the FFT card. For a 512 point fft, (M0=3) + (the additional 5) = a shift of 8 places. For the first fft in the 4 ms cycle this gives S.SSSSSSSS,FFFFFFFF,FFF. Adding this to itself 256 times (the number of spectral points for a 512 point fft) gives S.FFFFFFFF,FFF00000,000.

The above ascii drawing does not show initial phases to the fft card nco which are present for 1024 and 2048 point ffts.

2.2.3.1 OVLBI Considerations

In Orbiting Very Long Baseline Interferometry (OVLBI), the delay rates may vary rapidly enough that the delay may change by an integer value within a 4ms period. This can be compensated for by having the nco count through multiple cycles within an fft period.

For OVLBI, extra bits of delay can be accounted for by a toggling of the msb of the fft nco output (as referenced above). The sign bit is the msb. For the first fft cycle of a 4 ms cycle, the largest value the nco can count to is 0.FFFFFFFF,FFFFFFFF,FFF (before any R values are added in subsequent fft cycles). In this first fft cycle, the sign bit is zero, since the initial phase is positive. No additional bits delay can occur. This gives slightly less than 1 bit maximum delay. Adding in the R value for subsequent fft cycles allows the

exceeding of one bit delay.

Since the M0 shift is done before the FSTC Xilinx, it opens up some headroom. So now, when the R value is added to the initial phase, it can create a number larger than the initial F number. The amount of headroom increases with fft size as shown below.

Another way of looking at it is that for a 64 point fft, there is no M0 shift. The maximum positive value before the 5 place shift is 0.11111111,11111111,111 where the msb is the sign bit. This is the output of the initial phase plus the phase increment. The 5 place shift divides the number by 32 to give 0.00000111,1111111,111. Then the nco adds the number to the accumulator 32 times so no overflow can occur. The sum is 0.11111111,11111100,000. The sign bit does not transition once. This gives slightly less than 1 bit fstc delay. Similarly, the largest magnitude, 2's complement, negative number to survive the 5 bit shift is 1.00000000,00000100,000. After the 5 bit shift we obtain 1.11111000,00000000,001. Adding this to itself 32 times gives the original 1.00000000,00000100,000.

For a 128 point fft, the maximum value before the 5 place shift is still 0.11111111,11111111,111. Actually, this is the largest value for all fft sizes. The 5 place shift still divides the number by 32 to give 0.00000111,1111111,111. Now the nco adds the number to the accumulator 64 times, so overflows can occur. The total would be 1.11111111,11111000,000 only the msbit overflows. This allows slightly less than two extra bits delay. The largest magnitude negative number to survive the 5 bit shift is still 1.00000000,00000100,000. After the 5 bit shift we obtain 1.11111000,00000000,001. Only now, adding this to itself 64 times gives the 0.00000000,00001000,000. Tabulating the running sum, in hex for convenience: 1 1.F8002 2 1.F0004 3 1.E8006 ... 32 1.00040 33 0.F8042 34 0.F0042 35 0.E8046 ... 62 0.1007C 63 0.0807E 64 0.00080 Above, there are 2 sign changes per cycle for 2 bits delay.

Each subsequent fft size has twice the number of additions from the previous.

For a 512 point fft, the maximum positive value before the 5 place shift is still 0.11111111,11111111,111. The 5 place shift still divides the number by 32 to give 0.00000111,1111111,111. Adding this number, 256 times, gives 111.11111111,111. The msb or 1's bit changes 8 times for 8 bits maximum delay.

For 1024 ffts, the number of additions is doubled by doubling the number of fft pipelines doing additions.

For 1024 point ffts, there is an extra multiplying of the rate to the fft nco times two (see k027d02.blk in Volume II). Note the 5 place shift is done in the Xilinx. The multiplying of the rate times two is done in the Xilinx, by an extra register in series with the serial output. The sign extension associated with the 5 place shift is done in the pals 7E and 7F following the Xilinx.

For a 1024 point fft, assume the maximum positive value before the 5 place shift is still 0.11111111,11111111,111. The 5 place shift still divides the number by 32 to give 0.00000111,1111111,111. The multiply by 2 gives 0.00001111,11111111,110. The sign extension in the pal treats the first 1 as a negative sign and extends it. This can be seen on the mcc phase plots as the slopes reversing sign. The MCC phase plots are viewed by invoking the plot program on a pc connected to the appropriate MCC.

So now, assume for a 1024 point fft the maximum positive value before the 5 place shift is 0.01111111,11111111,111. The 5 place shift still divides the number by 32 to give 0.00000011,1111111,111. The multiply by 2 gives 0.00000111,11111111,110. Now the sign bit that is extended is correct as a 0.

Adding this number to itself 256 times give 111.11111111,11. Thus the 1's bit toggles 8 times. This gives the maximum bits delay as 8 bits. This is the same as the 512 point fft was, due to the limitation introduced by the multiply by two.

For a 1024 point fft, assume the maximum negative value before the 5 place shift is still 1.00000000,00000100,000. The 5 place shift still divides the number by 32 to give 0.00001000,00000000,001. The multiply by 2 gives 0.00010000,00000000,010. The sign extension in the pal treats the 0 to the right of the 1 as a positive sign, when we want a negative sign. This can be seen on the MCC phase plots as the slopes reversing sign.

Also, the above example, the multiply by two shift, shifted a zero into the lsb. Technically, a 1 should have been shifted into the lsb, since it is a negative number. However, since the above number is added to itself only 256 times, the error in the lsb never propagates to the 8 msbs used by the FSTC Roms.

So now, assume for a 1024 point fft the maximum negative value before the 5 place shift is 1.10000000,00000100,000. The 5 place shift still divides the number by 32 to give 0.00001100,00000000,001. The multiply by 2 gives 0.00011000,00000000,010. Now the sign bit that is extended is correct as a 1. This gives 0.11111000,00000000,010 or 0.f8004. Adding this number to itself 256 times give f8.004. Thus the 1's bit toggles 8 times. This gives the maximum bits delay as 8 bits. This is the same as 512 was.

Thus, the above explains why the 1k fft has a maximum delay range of +- 8 bits instead of the +-16 bits expected. The +-16 bits expected comes from extrapolating the results in the table below.

The 2048 point fft is similar to the 1k fft, only using 4 pipelines instead of two. Pipelines two and three continue with an initial phase equal to the ending phase of pipelines 0 and 1.

For 2048 point ffts, there is also an extra multiplying of the rate to the fft nco times two as in the 1024 point fft (See k027d02.blk in Volume II). We start with 0.11111111,11111111,111 as the maximum positive number. The 5 place shift divides it by 32. We multiply it by 2. Then add it 512 times, due to the addition being done in the upper and lower pipelines. This gives a net multiplication by 32 for a value of 11111.11111111,11111110,000. This would lead us to expect a maximum delay of 32 bits. However the same sign extension problem exists as with 1k, to divide the maximum numbers by two. Thus, we can expect a maximum delay of 16 bits.

Values after M0 Shift	FFT Size	Max Bits delay Correction in 4ms
S.FFFFFFFF,FFFFFFFF,FFF	64-POINT TRANSFORMS	1
S.SSSSSSSS,SSRRRRRR,RRR	RATE FOR 64-POINT FFT	
S.SFFFFFFF,FFFFFFFF,FFF	128-POINT TRANSFORMS	2
S.SSSSSSSS,SSSRRRRR,RRR	RATE FOR 128-POINT FFT	
S.SSFFFFFF,FFFFFFFF,FFF	256-POINT TRANSFORMS	4
S.SSSSSSSS,SSSSRRRR,RRR	RATE FOR 256-POINT FFT	
S.SSSFFFFF,FFFFFFFF,FFF	512-POINT TRANSFORMS	8
S.SSSSSSSS,SSSSSRRR,RRR	RATE FOR 512-POINT FFT	
S.SSSSFFFF,FFFFFFFF,FFF	1024-POINT TRANSFORMS	8
S.SSSSSSSS,SSSSSSRR,RRR	RATE FOR 1024-POINT FFT	
S.SSSSSFFF,FFFFFFFF,FFF	2048-POINT TRANSFORMS	16
S.SSSSSSSS,SSSSSSSR,RRR	RATE FOR 2048-POINT FFT	

2.2.4 Pulsar Model

As with fringe and delay model computation, pulsar model computation in the VLBA correlator is performed on 3 levels with the top level computation being done in the VME computer system.

The next level of pulsar model computation is done on the FFT control card in the floating point model generator. Pulsar phase models at this level are performed every 4-msec and require the evaluation of two polynomial equations for each FFT card bin (each bin processes four channels of 10 stations). The last level of pulsar model computation is done on the FFT Control Card (FCC) and employs a hardware implementation of a two coefficient model generator.

The form of the first polynomial that is evaluated by the FFT control card can be seen below;

P = P0 + P1T + P2T**2 + P3T**3 + P4T**4 + P5T**5

where:

P0	DOUBLE PRECISION (IEEE FLOATING POINT)
P1	DOUBLE PRECISION
P2	DOUBLE PRECISION
P3	DOUBLE PRECISION
P4	DOUBLE PRECISION
P5	DOUBLE PRECISION

12 32-BIT WORDS PER MODEL

The evaluation of this polynomial results in a value for the pulsar phase which when converted to a 64-bit fixed point number is of the form;

IIIIIIII,IIIIIIII,IIIIIIII,IIIIIIII.PPPPPPPP,PPPPPPPP,PPPPPPPP,PPPPPPPP

where;

IIIIIIII,IIIIIIII,IIIIIIII,IIIIIIII. IS THE INTEGER PART OF THE PULSAR PHASE

and

```
.PPPPPPPP,PPPPPPPP,PPPPPPPP,PPPPPPPP  IS THE FRACTIONAL PART OF THE PHASE
 ^--------------------^               20-BITS RETAINED
```

The integer part of the pulsar phase is discarded and the most significant 20 bits of the fractional phase are kept and used in the two coefficient polynomial model generator to produce the pulsar phase for the pulsar gate generator.

The two coefficient model requires the computation of a second polynomial to yield the rate of change for the pulsar phase to a resolution of 20-bits. The form of the second polynomial is seen below;

```
R = R0  +  R1T  +  R2T**2  +  R3T**3  +  R4T**4
```

where:

```
R0      DOUBLE PRECISION
R1      DOUBLE PRECISION
R2      DOUBLE PRECISION
R3      DOUBLE PRECISION
R4      DOUBLE PRECISION
------------------------
10 32-BIT WORDS PER MODEL
```

the result of this equation is converted to fixed point and the 20 MSBits retained.

The two 20-bit numbers (the fractional pulsar phase described earlier, and the rate of change of this number described above), are then used by logic on the FFT control card to generate a 10-bit pulsar gate phase every 16-usec. This phase is then stored in the FFT control card pulsar gate generator.

See Section 2.1.2.6 above and 2.3 below, for further descriptions of the pulsar gate generator.

The 20-bit fractional pulsar phase as computed is a binary number between 0 and 1. With a minimum pulsar period of 0.001 second (including any playback speed-up factor), the pulsar phase can change by no more that 0.016 cycles in one 16-usec FFT cycle. This assumes 1 msec as period of the fastest pulsar. The precision used in the two coefficient pulsar phase model generation are seen below;

```
  0.FFFFFFFF,FFFFFFFF,FFF   FRACTIONAL PULSAR PHASE CALCULATED BY MODEL
+ S.RRRRRRRR,RRRRRRRR,RRR   MAX RATE
```

The initial phase is positive, since the integer part has been discarded from a positive number. The rate can be negative. The 10 most significant bits are used in the ram address.

2.2.5 Coefficient Offset and Scaling (Fringe, Delay, and Pulsar)

The VLBA VME computer system will offset all sets of polynomial coefficients passed to the FFT control card processor such that when a new set of parameters becomes active on a predetermined time tick, the appropriate time for each station is used.

The FFT control card floating point processor will update all of the two parameter model coefficients every 4.096 msec model update cycle. This involves evaluating the appropriate 6-term magnitude and 5-term rate polynomials and passing new estimates of magnitude and rate to the two parameter model generators. Parameter scaling of the polynomial coefficients are such that the FFT control card floating point processor uses a time increment of 1 for a 4.096

msec correlator model update cycle.

In addition to the time scaling, the VME computer does magnitude scaling of the fringe parameters. The FFT control card does a floating point to fixed point conversion on the 6-term fringe phase polynomial result, and on the 5-term fringe rate polynomial result. The 64-bit fixed point products are of the form;

```
FFFFFFFF,FFFFFFFF,FFFFFFFF.FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF
.SSSSSSSS,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR
```

in 64-bit fixed point format. The arbitrary placement of the decimal point determines what magnitude scaling is required. Magnitude scaling factors of 2**40 for the phase polynomial coefficients, and of 2**64 for the rate polynomial coefficients are thus required.

Before the phase polynomial coefficients are scaled by 2**40 as above, the polynomial must be evaluated for the first value of time applied in the new model period. The 24 bit integer portion of the 64-bit fixed point result must then be subtracted from the P0 coefficient. Subtracting as specified means the 24 bit integer portion will start out the model period at zero. This is intended to insure that the integer turns accumulated by the hardware model generator does not exceed a 23-bit (8.38 X 10**6 turns) range during the 60 second update period (wall clock time). As can be seen in the last two bit expanded expressions above, there is only 23-bits of head room for the integer turns component of the fringe model after evaluation of the fringe phase polynomial (the 24th bit is the sign bit which may be positive or negative).

As of 5-6-93, the following worst case fringe rate values have been estimated for ground based and space based observations:

Case	Ground	Space
Maximum Fringe Rate	140 kHz	850 kHz
Maximum Turns in 120 s Period	1.68 E7	1.02 E8
Maximum Period for 8.38e6(2**23) Turns	59.9 s	9.86 s

As can be seen, even for ground based observations, the worst case needs to be guarded against.

Magnitude scaling by the VME computer for the delay model is such that when the FFT control card does a floating point to fixed point conversion on the 6-term delay polynomial result, and on the 5-term rate polynomial result, the 64-bit fixed point results are of the form;

```
00000000,00000000,IIIIIIII,IIIIIIII.FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF
S.SSSSSSS,SSRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR,RRRRRRRR
```

in 64-bit fixed point format. The delay result above is then divided at the MS/LS word boundary with 16-bits of integer delay being sent to the deformater card and 20-bits of the fractional delay and 20-bits of the fractional delay change rate being sent to the two coefficient model generator. Magnitude scaling factors of 2**32 for the delay polynomial coefficients and of 2**63 for the rate polynomial coefficients are thus required.

Magnitude scaling by the VME computer for the pulsar model is such that when the FFT control card does a floating point to fixed point conversion of the 6-term pulsar phase polynomial result and on the 5-term rate polynomial result, the 64-bit fixed point results are of the form;

```
IIIIIIII,IIIIIIII,IIIIIIII,IIIIIIII.PPPPPPPP,PPPPPPPP,PPPPPPPP,PPPPPPPP
00000000,00000000,00000000,00000000.00000RRR,RRRRRRRR,RRRRRRRR,RRRRRRRR
```

in 64-bit fixed point format. Magnitude scaling factors of 2**32 for the phase polynomial coefficients and of 2**32 for the rate polynomial coefficients are thus required.

2.2.6 Parameter Storage (Per FFT Control Card)

MODEL	STATIONS	CHANNELS OR PHASE CENTERS	PARAMETERS	TOTAL 32-BIT WORDS
FRINGE PHASE	10	4	6	480
FRINGE RATE	10	4	5	400
TIME VALUES	11	1	1	22
DELAY	10	2	7 (INC M0)	280
FRACT DELAY RATE	10	2	5	200
PULSAR PHASE	-	1	6	12
PULSAR RATE	-	1	5	10
				1404

The parameter memory on the FFT control card has storage capacity for 4 complete sets of model parameters, where one set is shown above. This storage capacity provides for four banks of two model sets. Model switching can be done between 2 active sets while parameters in the inactive bank are being updated by the VME computer. See Appendix III for the exact parameter memory allocation.

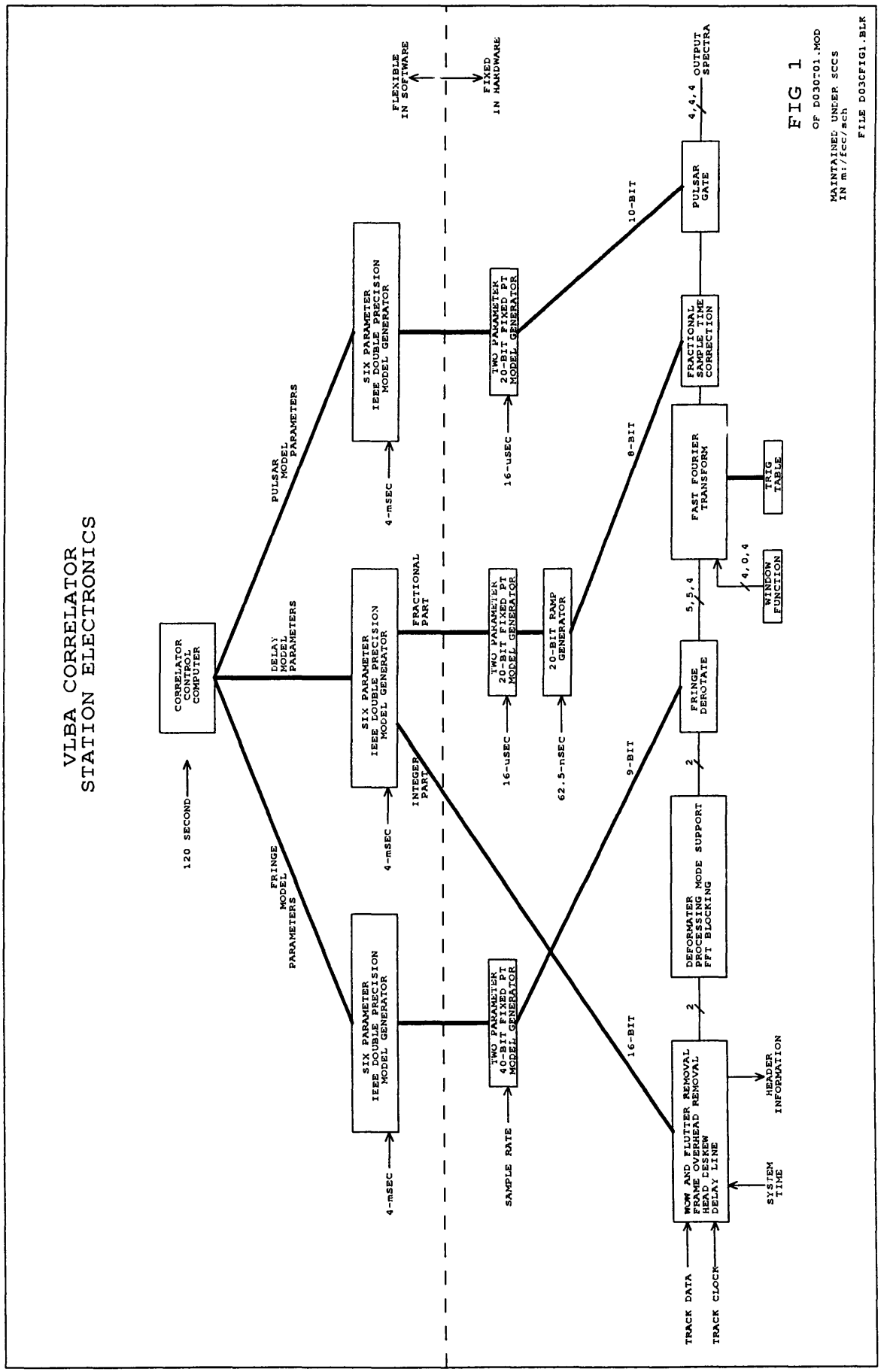
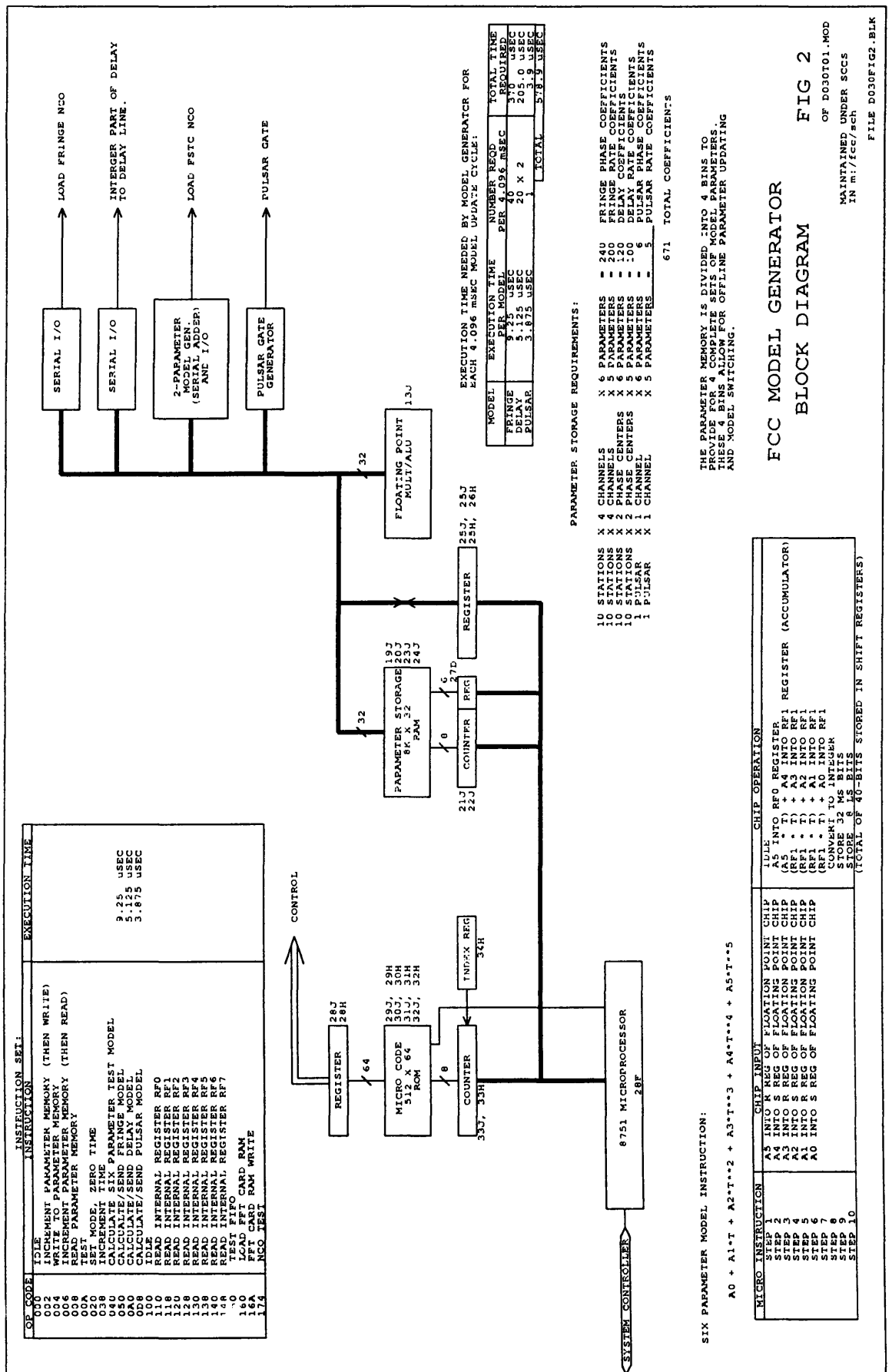


Figure 1 VLBA Correlator Station Electronics Block Diagram



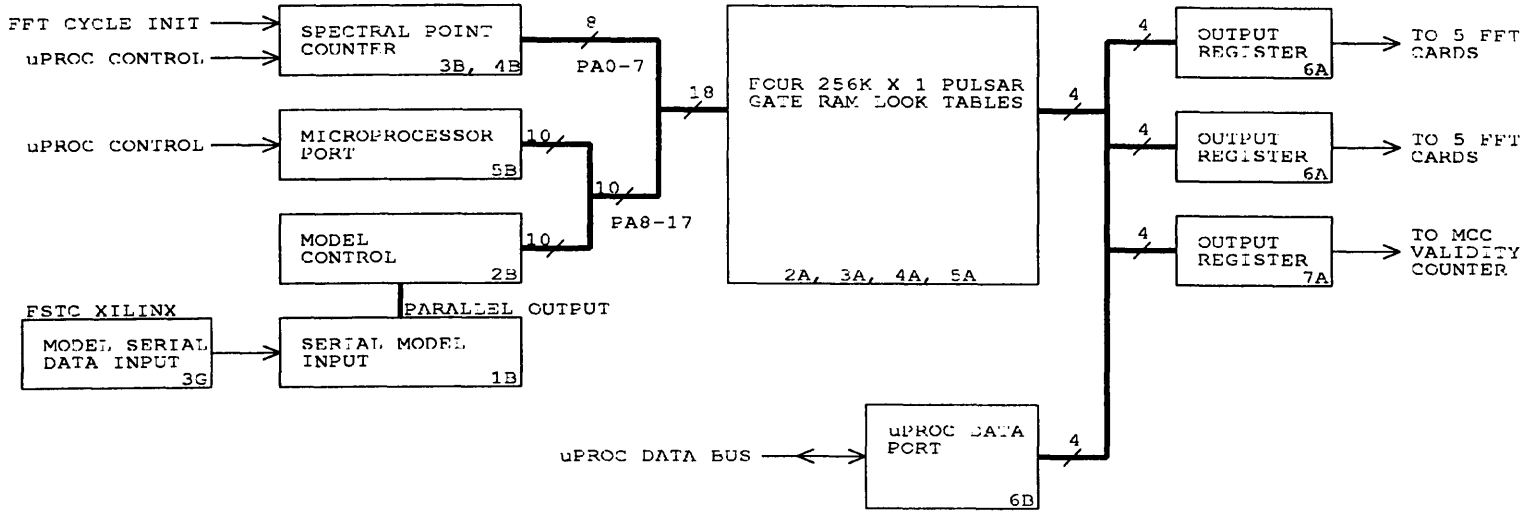
OP CODE	INSTRUCTION	EXECUTION TIME
000	IDLE	
002	INCREMENT PARAMETER MEMORY (THEN WRITE)	
004	WRITE TO PARAMETER MEMORY	
006	INCREMENT PARAMETER MEMORY (THEN READ)	
008	READ PARAMETER MEMORY	
00A	TEST	
020	SET MODE, ZERO TIME	
038	INCREMENT TIME	
040	CALCULATE SIX PARAMETER TEST MODEL	9.25 uSEC
050	CALCULATE/SEND FRINGE MODEL	5.125 uSEC
068	CALCULATE/SEND DELAY MODEL	5.125 uSEC
088	CALCULATE/SEND PULSAR MODEL	3.875 uSEC
100	IDLE	
110	READ INTERNAL REGISTER RF0	
118	READ INTERNAL REGISTER RF1	
120	READ INTERNAL REGISTER RF2	
128	READ INTERNAL REGISTER RF3	
130	READ INTERNAL REGISTER RF4	
138	READ INTERNAL REGISTER RF5	
140	READ INTERNAL REGISTER RF6	
148	READ INTERNAL REGISTER RF7	
[illegible]	TEST FIFO	
160	LOAD FFT CARD RAM	
16A	FFT CARD RAM WRITE	
174	NOO TEST	

MODEL	EXECUTION TIME PER MODEL	NUMBER REQD PER 4.096 mSEC	TOTAL TIME REQUIRED
FRINGE	9.25 uSEC	40	370 uSEC
DELAY	5.125 uSEC	20 X 2	205.0 uSEC
PULSAR	3.875 uSEC	1	3.9 uSEC
		TOTAL	578.9 uSEC

10 STATIONS	X 4 CHANNELS	= 240 FRINGE PHASE COEFFICIENTS
10 STATIONS	X 4 CHANNELS	= 200 FRINGE RATE COEFFICIENTS
10 STATIONS	X 2 PHASE CENTERS	X 6 PARAMETERS = 120 DELAY COEFFICIENTS
10 STATIONS	X 2 PHASE CENTERS	X 5 PARAMETERS = 100 DELAY RATE COEFFICIENTS
1 PULSAR	X 1 CHANNEL	X 6 PARAMETERS = 6 PULSAR PHASE COEFFICIENTS
1 PULSAR	X 1 CHANNEL	X 5 PARAMETERS = 5 PULSAR RATE COEFFICIENTS
		671 TOTAL COEFFICIENTS

MICRO INSTRUCTION	CHIP INPUT	CHIP OPERATION
STEP 1	A5 INTO K REG OF FLOATING POINT CHIP	IDLE
STEP 2	A4 INTO S REG OF FLOATING POINT CHIP	A5 INTO RF0 REGISTER
STEP 3	A3 INTO S REG OF FLOATING POINT CHIP	(A5 * T) + A4 INTO RF1 REGISTER (ACCUMULATOR)
STEP 4	A2 INTO S REG OF FLOATING POINT CHIP	(RF1 * T) + A3 INTO RF1
STEP 5	A1 INTO R REG OF FLOATING POINT CHIP	(RF1 * T) + A2 INTO RF1
STEP 6	A0 INTO S REG OF FLOATING POINT CHIP	(RF1 * T) + A1 INTO RF1
STEP 7		(RF1 * T) + A0 INTO RF1
STEP 8		CONVERT TO INTEGER
STEP 9		STORE 24 MS BITS
STEP 10		STORE 16 LS BITS
		(TOTAL OF 40-BITS STORED IN SHIFT REGISTERS)

Figure 2 FCC Model Generator Block Diagram



The pulsar gate generator RAM is programmed with the pulsar state. RAM chips 2A, 3A, 4A, and 5A can be thought of as a three dimensional array of bits where each bit gives the on/off state of the pulsar for the three conditions of pulsar phase, spectral point, and channel (actually, each bit can be more accurately described as providing integrate/don't integrate information).

The pulsar phase dimension, provided by RAM address bits PA8 thru PA17, gives the phase of the pulsar being observed as specified by the model generator. Every 4 msec, the floating point model generator computes the pulsar phase and the rate of change of this phase from the model parameters supplied by the real time computer system. These two numbers are stored in the Xilinx 2-parameter model stage where they are used to compute the 10 bit PA8 thru PA17 RAM address every FFT cycle. Thus, every 16-usec FFT cycle one of 1024 possible pulsar phase estimates ,as derived from the pulsar model, drives the RAM.

The spectral point RAM dimension, provided by RAM address bits PA0 thru PA7, in effect yields a different 1024 pulsar gate for each spectral point. This structure can accommodate frequency dispersion in the pulse.

The last dimension of the pulsar gate RAM is the 4-bit wide RAM output. This dimension is the channel dimension as each of the four RAM outputs provides the pulsar gate for a different FFT engine. When 1024- or 2048-point transforms are being performed, the 4-wide RAM output dimension reverts to the spectral point dimension.

FIG 3

OF D030T01.MOD

MAINTAINIED UNDER SCCS

IN m:/fcc/sch

FILE D030FIG3.BIK

A size

Figure3 Pulsar Gate Generator Block Diagram

2.2.7 Fringe Rotator Rom Contents Generation

The fringe rotator roms are located on each FFT card.

All files related to generating the fringe rotator rom contents are found in /vlbsoft/pal_prom/proms/fftfrng under SCCS.

Executing make_fringe compiles the program fringe.c, then executes it. This generates the two prom binary image files, named frngu108.ms and frngu107.ls (for "fringe" and the two U numbers on the FFT card). Duplicates of the roms serve the other three pipelines.

The address to the prom consists of a two bit sample and a nine bit angle index:

```
          ms------prom adr bits----------ls
          D1 D0  A8 A7 A6 A5 A4 A3 A2 A1 A0
          -- --  -- -- -- -- -- -- -- -- --
         |2 bit ||    9 bit fringe angle  |
         |sample||                        |
         |______||________________________|
```

The prom data output is a 7,7,2 format number, which is connected to the 7,7,4 input of the VLBA1 at stage 0 of a FFT pipeline:

(bits 0 - 15 are prom output bits, bits 0 - 17 are 7,7,4 bus)

```
     |    frngu108.ms        |     frngu107.ls       |
     |                       |                       |
17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 gnd |el e0|i6 i5 i4 i3 i2 i1 i0|r6 r5 r4 r3 r2 r1 r0|
     |     | s.                  | s.                  |
     |2 bit|                     |                     |
     |exp  | 7 bit imag          | 7 bit real          |
     |_____|_____________________|_____________________|
```

The 9 bit angle index is the result of truncating the 40 bit NCO value. Thus, for a given 9 bit index value, the actual angle required may be any value between the index value and the next higher 9 bit index value. We take the value half way between.

Since 9 bits is 512 states, we use a 1024 entry angle table.

NCO index 0-511	angle table index 0-1023	
	0	As indicated, for each 9 bit
0	1	index from the NCO, we need
	2	to look up the rotation angle
1	3	associated with an angle table
	4	index equal to:
2	5	
	6	(2 x NCO index) + 1
3	7	
508	1017	
	1018	
509	1019	
	1020	
510	1021	
	1022	
511	1023	

For 7 bit mantissas, we need a factor of (2^n - 1)/(2^n) where n is 6 for 6 bits of magnitude, to limit the maximum magnitude to 0.111111. Thus we need a factor of (63/64).

The two bit sample is defined as follows:

		D1	D0	(D1,D0) sample index	weight factor
	+n=	1	1	index=3	+1.0
+Vth	.....				
	+1=	1	0	index=2	+1.0 / 3.3358750
0V	_____				
	-1=	0	1	index=1	-1.0 / 3.3358750
-Vth	.....				
	-n=	0	0	index=0	-1.0

Given the NCO index and the sample index, the equations for the real and imaginary portions of the 7,7,2 value are:

real= (63/64) * weight[sample index} * cosine[((2pi)/1024)*(2*NCOindex +1)]

imag= -(63/64) * weight[sample index} * sine[((2pi)/1024)*(2*NCOindex +1)]

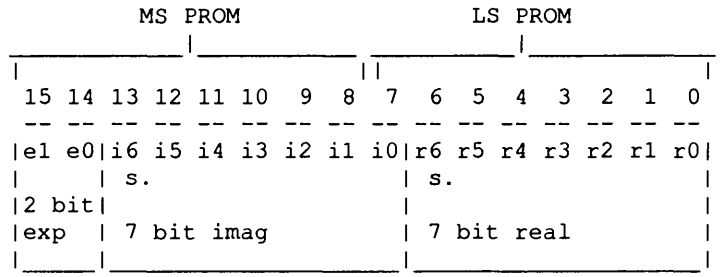
so that we have a complex result of the form cos(theta) + j sin(theta).

In the process of converting from floating point to fixed point values, we normalize the values, keeping in mind that there is a common exponent for the

real and imag parts.

The real and imaginary mantissas are normalized such that the MSB of the larger value is always a one.

The final data tables are programmed into the two 8 bit wide PROMs:

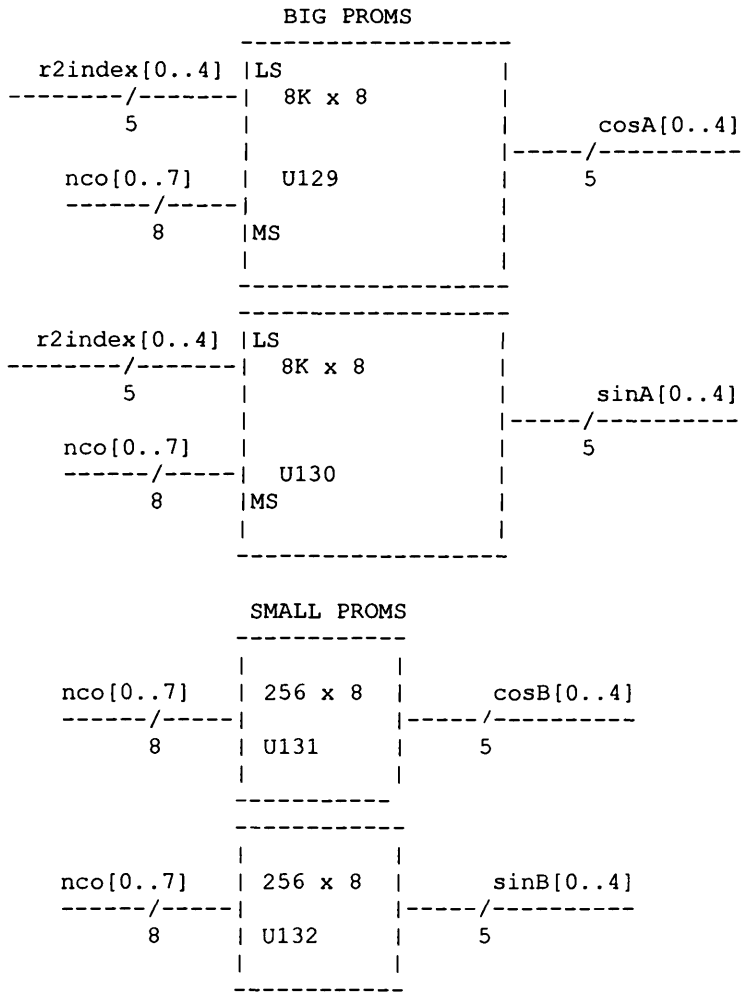


2.2.8 FSTC Rom Contents Generation

The FSTC roms are located on each FFT card. They are duplicated for each of the four pipelines.

All files related to generating the FSTC rom contents are found in vlbsoft/pal_prom/proms/fftfstc/SCCS. Executing the script make_fstc compiles the program fstc.c, then executes it to generate the four binary image files defined below.

There are four proms in a set (two "BIG PROMS" and two "SMALL PROMS"). A 5 bit index is presented to each big prom, along with the 8 bit nco rotation index. The MSB of the five bit index represents 90 degrees. Only the nco rotation index goes to the small proms.



The U numbers shown are for the upper pipeline on the FFT card. Binary file images of the proms are generated under the following file names:

u129cosa.fst
u130sina.fst
u131cosb.fst
u132sinb.fst

The big proms are only used when stage 5 does a radix 2 fft. That is for FFT sizes 128, 512 and 2K. For FFT sizes of 128 and 512, the r2index represents from 0 to less than 180 degrees. For a fft size of 2K, the r2indices to pipelines 0 and 1 represents from 0 to less than 90 degrees. The r2indices for the lower pipelines are formed by setting the msb=1, to give from 90 to less than 180 degrees.

The following table indicates the phase rotation error in degrees caused by a delay error. This error in the output spectral points is removed by the FSTC correction.

nco bits 7654 3210	offset delay (bits)	phase rotation @ high band edge (degrees)	twiddle correction (degrees)
0000 0000	0.0	0	0
0100 0000	+0.5	+90	-90
1000 0000	+1.0	+180	-180
1100 0000	+1.5	+270	-270
1111 1111	<+2.0	<+360	>-360

The master table of angles used for generating twiddles is used as the source of corrected twiddles for the FSTC proms. This table contains the '550' format sines and cosines for each twiddle from 0 through 2047. The sine value for each twiddle index has been negated so that the table stores cos(theta) -j sin(theta). Thus the values in the table at twiddle index 512 are the values for minus 90 degrees, the values at twiddle index 1536 are for plus 90 degrees etc. In other words, moving counter clockwise through this table (as from index 0 to 1 to 2 etc.) is a negative rotation and moving clockwise is a positive rotation.

In order to cancel the phase rotation error due to a delay error, we need to add a correction to the twiddle factors. The correction needs to be equal in degrees but opposite in sign to the error. Thus, for example, with the nco bits = 0100 0000 (a phase rotation error of +90 degrees), we need to correct the twiddle by -90 degrees.

The values in the master table are stored as both rounded down (value) and rounded up (value+1) values, along with counts of how many of the 128 angle tables need each of the two values. (See the definition of "struct master" in Section 2.2.9.3, and all of Section 2.2.9 for more details.)

To extract a sine or cosine from the master table, take the (value + 1) entry if the number of tables needing (value + 1) is greater than 63, otherwise take the rounded down value. (If 64 or more tables were specified to need the rounded up value, then the msb of the round field was high.)

For the small proms, the following table indicates which twiddle to extract from the master table for each of the nco bit patterns: (determined by finding the number of indices to rotate plus or minus from zero)

twiddle index to extract = nco*8

nco count	nco bits 7654 3210	correction angle (degrees)	x= (nco*8) nr of indices to rotate	x twiddle index to extract	
0	0000 0000	0	0	0	clockwise
1	0000 0001		+8	+8	rotation
	|			|	for positive
62	0011 1110		+496	+496	angle
63	0011 1111		+504	+504	
64	0100 0000	-90	+512	+512	counter-
65	0100 0001		+520	+520	clockwise
	|			|	rotation
128	1000 0000	-180	+1024	+1024	for negative
	|			|	angle
192	1100 0000	-270	+1536	+1536	
	|			|	
255	1111 1111	-360-	+2040	+2040	-360- means 1 lsb > than -360

Every eighth twiddle index is used (2048/256 = 8).

For the big proms, each integer value represented by the five bits of r2index[0..4] is formed from the most significant 5 bits of the twiddle number of the angle required. The lower bits will have been used to make multiple tables, similar to the 128 multiple tables for the other stages. Based on the twiddle step below, the number of multiple tables possible is determined by the number of "x" in the lower bits. Thus, there will be two sets of 64 identical tables for FFT size 128, eight sets of 16 identical tables for FFT size 512 and four sets of 32 identical tables for FFT size 2048.

				9	8	7	6	5	4	3	2	1	0	
FFT size	twiddle step	max twiddle	#bflys	512	256	128	64	32	16	8	4	2	1	nr of tables
128	16	1008	64	x	x	x	x	x	x	0	0	0	0	2
512	4	1020	256	x	x	x	x	x	x	x	x	0	0	8
2048	1	511	1024	x	x	x	x	x	x	x	x	x	x	32

The following table shows the twiddle index associated with each r2index:

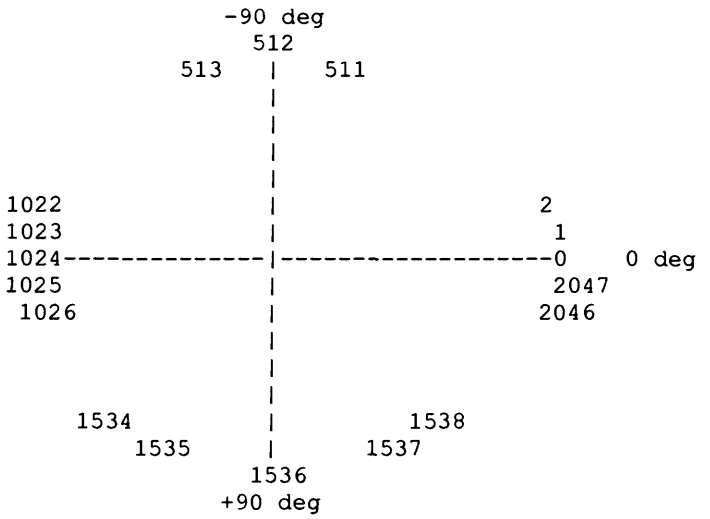
r2 ind val	r2index 4	3	2	1	0	twiddle angle (degrees)	twiddle index	
31	1	1	1	1	1	-180-	992	-180- means 1 lsb > than -180
30	1	1	1	1	0		960	
29	1	1	1	0	1		928	
|		|					|	
18	1	0	0	1	0		576	
17	1	0	0	0	1		544	
16	1	0	0	0	0	-90	512	
15	0	1	1	1	1	-90-	480	
14	0	1	1	1	0		448	
|		|					|	
1	0	0	0	0	1		32	
0	0	0	0	0	0	0	0	

Every 32nd twiddle index is used (512/16 = 32) since four bits of magnitude is used to represent the angles between 0 and 90 degrees.

The nco bits are now used to apply a rotation to the twiddle index from the above table. Add the twiddle index to the rotation offset to find the twiddle index to look up in the master table. Thus the final relation is:

twiddle index to extract = (r2index*32 + (nco*8)) % 2048

The following is for visual reference: (cos(theta) -jsin(theta))



2.2.9 Trig Tables Generation

2.2.9.1 Introduction

The angle tables are produced by software in the directory: /home/magnolia2/vlbsoft/fftangles.

The makefile generates angle tables for the FFT pipelines to be down-loaded to the FFT Control Card. Multiple tables are used, 128 tables total per stage. By using multiple tables, it is possible to synthesize twiddle factors of nigher precision than the 8 to 10 bits used. The average of the tables is the more precise value.

The FFT Control Card stores all tables in rams. For stages 2 and 3, the storage provided actually has room for 256 tables, but we ignore the extra capacity and assume each stage has only 128 tables.

2.2.9.2 Butterfly List

The following table is the first portion of a listing of butterflies for each stage of a 2K FFT pipeline. The actual order of points into the butterfly portion of the VLBA1 have the middle two points reversed. Thus, the first butterfly of stage 1 takes points in the order 0,8,4,12 rather than 0,4,8,12. (The point indices are the "in place" ram indices.)

STAGE 0		STAGE1		STAGE2		STAGE3		STAGE4		STAGE5	
0--	0.00	0--	0.00	0--	0.00	0--	0.00	0--	0.00	0--	0.00
1--	0.00	4--	0.00	16--	0.00	64--	0.00	256--	0.00	1024--	0.00
2--	0.00	8--	0.00	32--	0.00	128--	0.00	512--	0.00	1--	0.00
3--	0.00	12--	0.00	48--	0.00	192--	0.00	768--	0.00	1025--	0.18
4--	0.00	1--	0.00	1--	0.00	1--	0.00	1--	0.00	2--	0.00
5--	0.00	5--	22.50	17--	5.62	65--	1.41	257--	0.35	1026--	0.35
6--	0.00	9--	45.00	33--	11.25	129--	2.81	513--	0.70	3--	0.00
7--	0.00	13--	67.50	49--	16.88	193--	4.22	769--	1.05	1027--	0.53
8--	0.00	2--	0.00	2--	0.00	2--	0.00	2--	0.00	4--	0.00
9--	0.00	6--	45.00	18--	11.25	66--	2.81	258--	0.70	1028--	0.70
10--	0.00	10--	90.00	34--	22.50	130--	5.62	514--	1.41	5--	0.00
11--	0.00	14--	135.00	50--	33.75	194--	8.44	770--	2.11	1029--	0.88
12--	0.00	3--	0.00	3--	0.00	3--	0.00	3--	0.00	6--	0.00
13--	0.00	7--	67.50	19--	16.88	67--	4.22	259--	1.05	1030--	1.05
14--	0.00	11--	135.00	35--	33.75	131--	8.44	515--	2.11	7--	0.00
15--	0.00	15--	202.50	51--	50.62	195--	12.66	771--	3.16	1031--	1.23
16--	0.00	16--	0.00	4--	0.00	4--	0.00	4--	0.00	8--	0.00
17--	0.00	20--	0.00	20--	22.50	68--	5.62	260--	1.41	1032--	1.41
18--	0.00	24--	0.00	36--	45.00	132--	11.25	516--	2.81	9--	0.00
19--	0.00	28--	0.00	52--	67.50	196--	16.88	772--	4.22	1033--	1.58
20--	0.00	17--	0.00	5--	0.00	5--	0.00	5--	0.00	10--	0.00
21--	0.00	21--	22.50	21--	28.12	69--	7.03	261--	1.76	1034--	1.76
22--	0.00	25--	45.00	37--	56.25	133--	14.06	517--	3.52	11--	0.00
23--	0.00	29--	67.50	53--	84.38	197--	21.09	773--	5.27	1035--	1.93

Stages 0-4 are radix 4 butterflies, stage 5 is a radix 2. Stage 0 does not have a trig table. Stage 1 repeats angles every four butterflies (every 16 angles). Stage 2 repeats angles every 16 butterflies (every 64 angles). Stage 3 repeats angles every 64 butterflies (every 256 angles). Stage 4 repeats angles every 256 butterflies (every 1024 angles). Stage 5 has 1024 butterflies with 2048 angles total, where half of the angles are zero degrees.

For a 1K FFT, half of the butterflies are done in a single pipeline. For a 2K FFT, one quarter of the butterflies are done in a single pipeline.

For 1K and 2K FFTs, stage 3 butterflies are done in different sequences, depending on if the pipeline is an "upper" or "lower" pipeline. This is done so that stage 4 "upper" can receive all even numbered points, and stage 4 "lower" can receive all odd numbered points. This requires that trig tables to stage 3 upper and lower be different. As a result of this requirement, butterflies at stage 3 are always done in this manner, even for smaller size FFTs. The upper pipeline does butterflies in normal order (0,1,2,3 etc.). The lower pipeline does butterflies in reversed order (1,0,3,2 etc.).

Thus we need tables of angles for stage 1, stage 2, stage 3 upper, stage 3 lower, stage 4 upper and stage 4 lower. For stage 5, we need tables for fft sizes 128, 512 and 2K. For FFT size 2K, the angles to stage 5 pipelines 2 and 3 are a direct function of the angles to pipelines 0 and 1 (add 90 degrees to the angle at the corresponding lower numbered pipeline to get the angle for the higher numbered pipeline).

The "angles" stored for stage 5 are different from those of stages 1-4. Due to the fact that the actual sines and cosines going to the twiddle inputs of stage 5 come from the FSTC proms, we do not store actual sines and cosines in the angle tables on the FFT Control Card. Instead, we store an index (0-31) that represents the twiddle index (0-1023) of the un-corrected angle required

for each butterfly at stage 5. This index, and the FSTC nco 8 bit index, form the address to the "big proms" on the FFT card.

Recall that half the angles are zero for stage 5. The stage 5 angle tables for fft sizes 128, 512 and 2048 will contain only the "non-zero" angles for each butterfly (unlike the radix 4 tables which contain all four angles for each butterfly). Two twiddle buses drive the FFT card for stage 5. One has the angles for 128 and 512 FFT sizes multiplexed together. The other has the angles for a 2048 point FFT. It appears that "which comes first" is flexible for the 128/512 bus (R2CNT0 can be programmed to handle either 128/512 or 512/128). We build the table with 128 first, followed by 512. The 2K bus is not flexible. The "odd" angles must proceed the "even" angles. Thus the sequence is angle 1,0,3,2,5,4 etc.

2.2.9.3 Angle Table Structure

A master angle table is used, with an entry for each twiddle index from 0 thru 2047. Each entry is a structure of the following form:

```
struct master
{
  double        factor;    /* factor */
  unsigned int  twid;      /* twiddle */
  unsigned char sin;       /* sine, 550 format */
  unsigned char sin1;      /* sine value + 1 */
  unsigned int  sinv;      /* nr of tables with sin value */
  unsigned int  sinv1;     /* nr of tables with sin value + 1 */
  unsigned char cos;       /* cosine */
  unsigned char cos1;      /* cosine value + 1 */
  unsigned int  cosv;      /* nr of tables with cos value */
  unsigned int  cosv1;     /* nr of tables with cos value + 1 */
};
```

The definitions of "value" and "value + 1" are discussed later. This table is used as the source of all sines and cosines to be entered in the 128 tables for FFT stages 1-4. It is also used by the program fstc.c as the source of the sines and cosines that form the contents of the fstc proms.

The actual function we want is cos(theta) - j sin(theta), so the sign bit of all sine values is inverted.

A factor of 15/16 (0.9375) is used so that sin(90 degrees) is 0.1111 instead of 1.0000, since there are only four magnitude bits. Early tests used various factors, so file names included the factor. When 0.9375 became the only factor used, the makefile was changed to rename the end product files as described later.

2.2.9.4 Angle Table Creation

The program masterangle creates the master table and write the table to a file with the name derived from the factor.

The program bldstagesmo reads the master table and writes intermediate files for each stage as follows:

Stages 1-4:

Angle Table	Total rad4 bflys	#Bflys before repeat	#Angles before repeat	#Angles x #tables (128 tables)	File name for data file (128 tables)
stg 1	128	4	16	2048	stage1.ang
stg 2	128	16	64	8192	stage2.ang
stg 3 upper	128	64	256	32768	stage3u.ang
stg 3 lower	128	64	256	32768	stage3l.ang
stg 4 upper	128	128	512	65536 (even bflys)	stage4u.ang
stg 4 lower	128	128	512	65536 (odd bflys)	stage4l.ang

Stage 5:

Angle Table	Total rad2 bflys per pipe	#Angles	#Angles x #tables (128 tables)		File name for data file (128 files)
stg 5 128/512	256	512	65536	256 angles for each fft size (128 and 512)	stg5lt2k.ang
stg 5 2K	256	512	65536	256 angles for each of two pipes	stg5eq2k.ang

Note that the 256 angles for FFT size 128 consists of four sets of 64 angles.

The data files indicated in the tables above contain structures with the following layout:

```
struct angle
{
      unsigned short int twid; /* twiddle index */
      unsigned char sine;  /* 550 format */
      unsigned char cosine;  /* 550 format */
};
```

where the order of angles in the file corresponds directly to the order in which the angles must be fed to the FFT stage. Each file is prefixed with a header containing the double precision value of the factor that was used, followed by the unsigned int stage number.

Thus, the size of a file is the header = 8 + 4 = 12 bytes, plus the 4 bytes per angle entry (this is the size of struct angle).

2.2.9.5 Angle Table Ram Images

The data in the above files must be re-arranged to build files containing ram images of the data to be downloaded to the FFT Control Card. For stage 1 these are exact ram images. For other stages, two physical rams are interleaved due to speed requirements. The interleaving is transparent to the ram image data file. At stages 2 and 3, the interleaving requirement results in a total ram size equal to twice the required storage. The excess capacity is not being used. The following bit definitions are used for building the ram images:

RAMS FOR STAGES 3 and 4:

```
 7    6    5    4    3    2    1    0      upper rams
---- ---- ---- ---- ---- ---- ---- ----
|3 ls upr sin| | all five bits upr cos|     full upper cos, part upper sin
-------------- ------------------------

 7    6    5    4    3    2    1    0      middle rams
---- ---- ---- ---- ---- ---- ---- ----
|sp| |all 5 bits lwr cos    | |2ms uprsin| part upper sin, full lower cos
---- ------------------------ ------------

 7    6    5    4    3    2    1    0      lower rams
---- ---- ---- ---- ---- ---- ---- ----
|   3 spare  | |all 5 bits lwr sin    |     full lower cos
-------------- ------------------------
```

RAMS FOR STAGE 5:

```
 7    6    5    4    3    2    1    0      upper rams
---- ---- ---- ---- ---- ---- ---- ----
|3ls lt2k sin| |all 5 bits lt2k cos   |     full lt2k cos, part lt2k sin
-------------- ------------------------     (lt2k stands for less than 2k, or the
                                                 128/512 point fft values)

 7    6    5    4    3    2    1    0      middle rams
---- ---- ---- ---- ---- ---- ---- ----
|sp |ls  | 4 ls bits of    | |2 ms bits|   part lt2k sin
|   |eq  | eq2k cos        | |of lt2k  |   part eq2k cos (eq2k stands for equals 2k)
|   |2k  |                 | |sin      |   part eq2k sin
|   |sin |                 | |         |
---- ---- ------------------ -----------

 7    6    5    4    3    2    1    0      lower rams
---- ---- ---- ---- ---- ---- ---- ----
|3 spare bits | ms | 4 ms bits of eq2k|     part eq2k sin
|             |bit | sin              |     part eq2k cos
|             |eq2k|                  |
|             |cos |                  |
------------- ----- ------------------
```

For stages 3, 4 and 5 there are three sets of interleaved rams. On the FFT Control Card logic drawings (1013d02 and 1013d03 in Volume II), the upper bank of rams for stages 3 and 4 store angles for the upper FFT pipelines (pipelines 0 and 2). The lower bank has bits for the lower pipelines. The middle bank has a mixture of angle bits for both the upper and lower pipelines. Thus the terms upper and lower can be related both to the banks of rams on the logic drawing and to the upper and lower FFT pipelines.

For stage 5, the upper bank of rams store angles for FFT sizes of less than 2K (128 and 512). The lower bank has angles for FFT size of 2K. The middle bank has a mixture of angle bits for both.

2.2.9.6 Angle Table File Names

The file names in the following table may be related to the banks of rams on the FFT Control Card. The RAM ADR CODE in the table is taken from the FFT Control Card logic, and is also used as the HCB sub-function code for each of the indicated transfers. The decimal numbers in parentheses in the NR HCB TRANSFERS column is the range of values for the third byte in the HCB header.

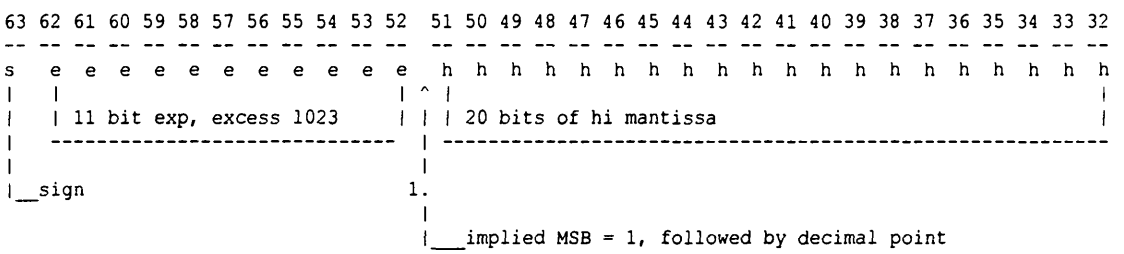
ORIGINAL FILE NAMES	RAM ADR CODE	TOTAL BYTES	NR HCB TRANSFERS (1024 bytes per)	FINAL SYSTEM FILE NAMES
stg1cos.ram	90	2048	2 (00-01)	m1cos.ram
stg1sin.ram	91	2048	2 (00-01)	m1sin.ram
stg2cos.ram	92	8192	8 (00-07)	m2cos.ram
stg2sin.ram	93	8192	8 (00-07)	m2sin.ram
stg3upr.ram	94	32768	32 (00-31)	m3upr.ram
stg3mix.ram	95	32768	32 (00-31)	m3mix.ram
stg3lwr.ram	96	32768	32 (00-31)	m3lwr.ram
stg4upr.ram	98	65536	64 (00-63)	m4upr.ram
stg4mix.ram	99	65536	64 (00-63	m4mix.ram
stg4lwr.ram	9a	65536	64 (00-63)	m4lwr.ram
stg5lt2k.ram	9b	65536	64 (00-63)	m5lt2k.ram
stg5mix.ram	9c	65536	64 (00-63)	m5mix.ram
stg5eq2k.ram	9d	65536	64 (00-63)	m5eq2k.ram

2.2.9.7 Generating Angles

For each twiddle index (0-2047), we first calculate the sine and cosine of the angle and multiply by a fractional factor. This factor must be less than or equal to 15/16 for 5,5,0 angles, so that the largest possible value is 0.1111 binary. The factor that is used is 15/16.

The double precision value must be converted to fixed point. A 16 bit integer will be produced from the double precision value as follows:

(lower 32 bits of double precision format not shown below)



Save the sign bit in a temporary variable. Subtract 1023 from the exponent and save the result in a temporary variable. The result should be less than zero, since the double precision value should be less than 1.0.

In the original value, set the sign=0 and set the exponent = 1. This places the "implied MSB" in the correct position with all higher bits set equal zero. Now shift the 64 bit word right a number of places equal to the result of the exponent - 1023.

Copy the saved sign bit back to the lsb position of the exponent field. Now, a final five place shift places the 16 bits of interest in bit positions 47 thru 32 for easy access.

```
63 62 61 60 59 58 57 56 55 54 53 52  51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32
-- -- -- -- -- -- -- -- -- -- -- --  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 0  0  0  0  0  0  0  0  0  0  0  1.  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h

                                    s  x  x  x  x  y  y  y  y  y  y  y  z  -  -  -

                                                 |                                              |
                                                 | final 16 bits end up here                    |
                                                 ------------------------------------------------
```

In order to generate 128 versions of each 5,5,0 angle table, we first convert to a 12,12,0 format in order to have 7 lower order bits to look at in order to see how many of the 128 tables receive "x" and how many receive "x+1", where x+1 is calculated by adding 1 to the x field below:

```
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
s . x  x  x  x  y  y  y  y  y  y  y  z  -  -  -
```

Bit z above is used to round off into the bits above z. The y field is then used as follows:

Enter the value "x" in 128-y tables.

Enter the value "x+1" in y tables.

Y	# of tables for X	# of tables for X+1
0	128	0
1	127	1
2	126	2
etc.		
126	2	126
127	1	127

For stage 5, the multiple tables are handled in a similar manner. The actual number of distinct tables is less than 128 and is a function of the FFT size. The following table indicates the method of determining the 5 bit index and number of distinct tables for each FFT size at stage 5.

128pt Bfly Nr	Twid Nr	5 2 1 Twid 1 5 2 6 3 1 2 6 8 4 2 6 8 4 2 1	512pt Bfly Nr	Twid Nr	5 2 1 Twid 1 5 2 6 3 1 2 6 8 4 2 6 8 4 2 1
0	0	0 0 0 0 0 0|0	0	0	0 0 0 0 0 0|0 0 0
1	16	0 0 0 0 0 0|1	4	4	0 0 0 0 0 0|0 0 1
2	32	0 0 0 0 0 1|0	8	8	0 0 0 0 0 0|0 1 0
3	48	0 0 0 0 0 1|1	12	12	0 0 0 0 0 0|0 1 1
4	64	0 0 0 0 1 0|0	16	16	0 0 0 0 0 0|1 0 0
5	80	0 0 0 0 1 0|1	20	20	0 0 0 0 0 0|1 0 1
6	96	0 0 0 0 1 1|0	24	24	0 0 0 0 0 0|1 1 0
7	112	0 0 0 0 1 1|1	28	28	0 0 0 0 0 0|1 1 1
8	128	0 0 1 0 0 0|0	32	32	0 0 0 0 0 1|0 0 0

Bfly Nr	2Kpt pipe0 Twid Nr	5	2	1								2Kpt pipe1 Twid Nr	5	2	1						
		1	5	2	6	3	1						1	5	2	6	3	1			
		2	6	8	4	2	6	8	4	2	1		2	6	8	4	2	6	8	4	2	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
1	2	0	0	0	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0	1	1
2	4	0	0	0	0	0	0	0	1	0	0	5	0	0	0	0	0	0	0	1	0	1
3	6	0	0	0	0	0	0	0	1	1	0	7	0	0	0	0	0	0	0	1	1	1
4	8	0	0	0	0	0	0	1	0	0	0	9	0	0	0	0	0	0	1	0	0	1
5	10	0	0	0	0	0	0	1	0	1	0	11	0	0	0	0	0	0	1	0	1	1
6	12	0	0	0	0	0	0	1	1	0	0	13	0	0	0	0	0	0	1	1	0	1
7	14	0	0	0	0	0	0	1	1	1	0	15	0	0	0	0	0	0	1	1	1	1
8	16	0	0	0	0	0	1	0	0	0	0	17	0	0	0	0	0	1	0	0	0	1

The ms 5 bits are the entry into the angle tables. The lower bits are used to specify how many tables receive "value" and how many receive "value+1" by padding on the ls end with zeroes to form a 7 bit number. This 7 bit number is used in the same manner as the 7 bit "y" field discussed earlier.

The resulting 5 bit values are then entered into both the sin and cos fields of the "angles" for stage 5.

2.2.9.8 Stage 1-4 tables

For stage?.ang files, the initial tables are built with all 128 versions of the tables identical, using only the sin and cos values. After this, for each twiddle and each sin and cos, the specified number of tables are updated with the sin+1 or cos+1 values.

The 16 angles for one of the 128 stage 1 tables, are the angles for four radix 4 butterflies. Thus the complete file for stage 1 is defined by:

```
struct angle stage1[128][4][4].
```

The definitions of the complete structures for each radix 4 stage are:

```
                          tbls bflys pnts
                          ---- ----- ----
struct angle  stage1 [128]  [ 4]  [4]
struct angle  stage2 [128]  [16]  [4]
struct angle stage3u [128]  [64]  [4]
struct angle stage3l [128]  [64]  [4]
struct angle stage4u [128] [128]  [4]
struct angle stage4l [128] [128]  [4]
```

There is a "twiddle number (twid)" associated with each butterfly. The actual twiddle values going into the butterfly are a function of:

```
0*twid       where the          0*twid
1*twid       actual order is:   2*twid
2*twid                          1*twid
3*twid                          3*twid
```

For stages 1, 2, 3 and 4 the twiddle starts at zero for butterfly number 0, and increments in the following steps to the indicated maximum twiddle (inclusive):

Stage	Twiddle Step	Twiddle Values
1	128	0,128,256,384
2	32	0,32,64,...480
3	8	0,8,16,...504
4	2	0,2,4,...510

Stage 3l, is the same as 3u with even/odd butterflies reversed (3u is normal order).

Stage 4u has the even butterflies for stage 4. Stage 4l has the odd butterflies.

2.2.9.9 Stage 5 Angles

Stage	FFT size	Twiddle Step	Maximum Twiddle
5	128	16	63*16 = 1008
5	512	4	255*4 = 1020
5	2K	1	511*1 = 511

The stage 5, FFT size 128 table has a twiddle step of 16, and a maximum twiddle of 1008. The angles for FFT size 128 are repeated four times (four sets of 64 angles), and are intermixed with the 256 angles for FFT size 512.

The stage 5, FFT size 512 table has a twiddle step of 4, and a maximum twiddle of 1020.

The stage 5, FFT size 2K table has a twiddle step of 1, and a maximum twiddle of 511, where the sequence of twiddles is 1,0,3,2,5,4 etc. Twiddles 512-1023 are "created" on the FFT card by adding 90 degrees to the lower twiddles.

2.2.9.10 Invoking Programs:

Running the Unix 'make' utility uses the Makefile to generate two sets of files:

m*.ram for use in the system.
t*.ram for use in the FFT Test Fixture.

The t*.ram files use 128 identical tables, instead of 128 different tables at each stage.

2.3 Overview of the Pulsar Gate Implementation

2.3.1 Introduction

This describes the function and requirements of the pulsar gate generator in the VLBA correlator. This generator divides a pulsar pulse cycle into 1024 parts and provides for independent save/discard qualifying of cross products across this range. Each spectral point has an independent 1024 entry save/discard mask and frequency dispersion in the pulse can be accommodated by programming phase offsets between the masks of adjacent spectral points.

The pulse phase is computed once, in the time it takes to accumulate a number of Nyquist rate samples equal to the FFT size. All samples in the FFT use this one pulse phase. The pulse phase resolution varies with the observation

parameters within a range, from 2-usec (for 64-point FFTs and a 16-MHz bandwidth), to 256-usec (for 2048-point FFTs and a 4-MHz bandwidth), and longer as the bandwidth goes down.

2.3.2 Hardware Description

The pulsar gate generator in the VLBA correlator is used to specify to the cross multipliers when the pulse of a pulsar is on or off (or, more accurately, when cross products should by integrated or discarded). The gates are applied at the output of the last stage asic in the fft pipelines. There are 4 pulsar gate generators in the correlator. Each generates a pulsar gate for the FFT cards, that process the signals of 4 channels from 10 stations. Each of the four generators are individually programmable and could, in principal, track 4 different pulsars. A more likely use for the multiple gates would be to take them in two sets of 2 (20-stations, 4-channels each). Different parts of the pulse could be integrated in different channels.

Each pulsar gate generator is a look-up table, stored in a 256K X 4 RAM. The 18-bit address range of the look-up table is broken up into two fields. The 8-bit field is nominally considered to be along the spectral point axis. The 10-bit field is along the pulsar phase axis. The 4-bit width of the RAM can be thought of as being along a channel axis. As will be seen below, however, the RAM address allocation changes with the correlator mode.

The spectral point index of the look-up table is complicated by the FFT size. Processing modes with FFTs of 512 points, resulting in 256 point spectra, are the simplest cases to explain. For this FFT size, the pulsar gate look-up table has a simple 8-bit index on the spectral point axis and a 10-bit pulsar phase axis (and a 2-bit channel axis).

For FFTs smaller than 512 points, the 8-bit index on the spectral axis traverses the spectra of several integral FFTs performed on contiguous samples. With the exception of 1024-point and 2048-point FFTs, station FFTs in the VLBA correlator are always performed over 512 consecutive Nyquist samples. In a 128-point FFT, for example, 512 consecutive (Nyquist) samples are blocked off into an FFT engine where, in essence, four 128-point FFTs are performed simultaneously. In this example, the pulsar gate look-up table, for a given pulsar phase, must do all the spectral point 0's for the four ffts, then all the spectral point 1's, etc.

For 1024 point FFTs, the look-up table structure changes from being 4 pulsar gate generators wide to being 2 wide. The lost factor provides the factor of 2 needed for the additional spectral points. Likewise, for 2048 point FFTs, the look-up table 4-channel index becomes an additional factor of 4 along the spectral point axis. In essence, what is happening is that for 1024 point FFTs, two FFT engines are connected together and, upon FFT completion, they output odd and even sets of 256 spectral points, respectively (for a total of 512). For 2048 point FFTs, four FFT engines are connected together and output the spectral points seen below;

FFT engine 0	even spectral points 0 thru 511
FFT engine 1	odd spectral points 0 thru 511
FFT engine 2	even spectral points 512 thru 1023
FFT engine 3	odd spectral points 512 thru 1023

2.3.3 Pulsar Model

The phase of a pulsar is tracked in the VLBA correlator by computing mathematical models of the pulse phase in software. This model computation is performed on 3 levels with the top level computation being done in the real time computer system (the VME based RTS) using CALC. Model parameters are

computed every 120 seconds of observe time at this level.

See Section 2.2.4 for a detailed description of the Pulsar Model.

Proper parameter scaling has the pulsar model run at the processing rate. This requires consideration of several factors including:
The transport speed-up factor during playback.
The effective rate of processing.
(which is determined by FFT overlap factor, oversample factor, FFT zero padding, etc).

2.3.4 Look-Up Table Contents

The pulsar gate look-up table RAM must be filled before a pulsar observation is processed on the VLBA correlator. As described above, each of the RAMs in a look-up table is 256K X 4 in size. The RTS will down-load the suitable files to fill the RAMs.

To generate a file for the observation, an astronomer must decide which phases of the pulse should be retained in the integrator and which phases should be discarded. A logic one in the RAM, for a given phase and spectral point number, will exclude the cross multiplication from the integration results.

The RAM essentially has one pulse cycle addressed across a 10-bit field of its storage space. The spectral point field of the RAM address is more complicated. The table below defines the spectral point RAM addresses and the variations with FFT size. The time resolution given in this chart is processing time. The effective time delay between adjacent FFTs on the RAM spectral point axis is the time required to gather a FFTs worth of samples at the Nyquist sample rate (corrected for any playback speed-up). This time, expressed in pulsar periods, must be used to offset later pulsar phase entries in the RAM.

FFT size	FFTs across RAM spectral point axis	time resolution of pulsar phase
64-point	8	2-usec
128-point	4	4-usec
256-point	2	8-usec
512-point	1	16-usec
1024-point	0.5 (1 across 2-bits of 4-bit wide RAM)	32-usec
2048-point	0.25 (1 across 4-bits of 4-bit wide RAM)	64-usec

To fill in the RAM file spectral point dimension, the astronomer must take the pulse dispersion into account as well. The spectral points come out of the FFT engine in frequency order. For short FFTs, first all the spectral point 0's are done, then all the spectral point 1's, etc.

2.4 FCC Utility Displays

This section describes operational procedures to aid in troubleshooting and monitoring the FCC, using the FCC microprocessor console.

The FCC HELP screen can be summoned by issuing the ? command on the FCC console.

See Appendix III for a complete mapping of the FCC memory.

The command D7D00 displays the active model. The active model for the ten stations and the pulsar are displayed as the values through 7D0A. There are

four models 0 through 3. Each model can be loaded with delay, fringe, and time parameters. The active model is the one currently being used.

The microprocessor command PMxxxx displays the contents of parameter memory, where xxxx is the address. Values in parameter memory are stored as 64 bit, IEEE floating point numbers. Appendix III gives the parameter memory addresses for the different models.

For example, suppose the active model is 0. PM0160 shows the times for the stations. Subsequent PM0160 commands should show the time having increased in value, if the FCC is observing.

As another example, suppose the active model is still 0. PM0800 shows the delay model parameters for station 0, delay center 0. This includes the phase and rate polynomial coefficients, plus the mode dependent shift parameter, as defined in Appendix III.

Fringe and Pulsar model coefficients can also be viewed in parameter memory.

3.0 The FFT Card

3.1 VLBA1 The Custom Application Specific Integrated Circuit (ASIC)

3.1.1 Introduction

The heart of the FFT Card is the VLBA1. This chip has several digital signal processing functions, the principal applications being in performing radix 4 or radix 2 FFT butterfly operations, and in performing floating point, complex multiply-accumulations.

A block diagram of the ASIC is k001d01.sch in Volume II.

3.1.2 Digital Signal Representation

The signal representations described in the ASIC and on the FFT card are generally expressed in complex, floating point form. The complex, floating point signal representation to be used is a non-standard form in which the real and imaginary components of a number have a common exponent bit field. Such complex numbers are expressed in a short hand fashion, an example of which is 774 indicating 7-bit real and 7-bit imaginary mantissa fields and a common 4-bit exponent field.

Each 7 bit mantissa is a sign magnitude number with an implied decimal point on the left. For example 1100000 represents -.100000 in binary or -.5 in decimal. When 4 bit mantissas are used as in 444 numbers, the three lsb's are truncated. For example: 1100 is -.5 for a 4 bit mantissa.

The 4 bit exponent is a number 0 to 12 with an implied negative sign. For example: The 774 number Real Mantissa 1100000, Imaginary Mantissa 0010000, Exponent 0010, represents real -.5, imaginary .25, with a common exponent of 2**-2 or 1/4. This gives: real -0.125, imaginary 0.0625 .

The exponent adjust function in the ASIC adds 0, 1, 2, or 3 to the implied negative exponent to keep it from becoming too negative. For example, if the exponent adjust is 1, the above number would become real -0.25, imaginary 0.125. Exponent adjust is set in the ASIC control word.

Five basic precisions are required in the various applications of this ASIC:

3.1.2.1 774 Numbers

Data points being Fourier transformed, input via the Normal or Auxiliary data port:

required precision	comment
774	Requires two 512 X 18 RAM banks.

3.1.2.2 504 Numbers

The Window Function, used in FFT butterfly Stage 0, and input via the twiddle port:

required precision	comment
Positive 504	(8 bits. Since the sign is always positive, it is not counted towards the bit total.) Used in the twiddle input port of the ASIC as follows: bits 0-3,5-8,with bits 4 & 9 externally grounded.

3.1.2.3 550 Numbers

The Sin/Cos twiddle factors, used in FFT butterfly Stages 1-5:

required precision	comment
550	Uses all 10 twiddle input bits.

3.1.2.4 444 Numbers

Points being multiply/accumulated in the MAC card. 444 Numbers are obtained by dropping the three lsbs from the mantissas of the 774 numbers output from the FFT Card.

acceptable precision	comment
444	Uses all 10 twiddle input bits plus shares pins NCOSI and NCOCI for input into the MAC Card ASICs.

3.1.2.5 15,15,6 Numbers

Accumulator precision (Used on the MAC card):

acceptable precision	comment
15,15,6	Requires a 36 bit wide RAM.

3.1.3 ASIC Functional Description

The ASIC used in the VLBA correlator is a multi-function chip that can be used for the following applications:

3.1.3.1 Radix 4 FFT butterfly

The ASIC must perform a radix 4, Decimation In Time (DIT), FFT butterfly. The butterfly is performed on four digitally sampled, complex data points clocked serially into the chip in four consecutive clock cycles. Three

input ports exist into the ASIC. There are normal and auxiliary ports for the points being transformed. These allow alternate data paths, used to allow 1024 or 2048 point ffts. The third port is for the FFT twiddle factors. One point and a twiddle factor enter the chip simultaneously, i.e. on the same clock edge. The ASIC is fully pipelined, so that one complex data point enters, and one complex data point exits the chip every 32 MHz clock cycle. In order to avoid offset biases that would result from using two's complement arithmetic, the points into and out of the chip are expressed in sign-magnitude floating point (774 Numbers). The internal fixed point adders work in one's complement arithmetic to avoid bias. RAM storage for the data points (not the twiddle factors), inserted between the chip input ports and the butterfly input circuitry, is required in all butterfly stages. The RAM addresses can be generated in the ASIC, or provided externally. A RAM configuration of two independently addressable 512 X 18 banks is required to allow double buffering of the points to be transformed.

3.1.3.2 Radix 2 FFT butterfly

The ASIC must also perform a radix 2, DIT, FFT butterfly on two digitally sampled, complex data points. The throughput is one complex data point per clock cycle. Most of the details defined above for the radix 4 function are true for this application.

3.1.3.3 Bypass function

The ASIC allows all points to flow straight through the chip unaltered except for a possible rearrangement in time sequence. This allows smaller fft sizes.

3.1.3.4 Complex Multiply/Accumulator

The ASIC is also used in the Multiplier Accumulator (MAC) Card. The ASIC first inputs two floating point, complex numbers, one on the butterfly data point input port and one through the twiddle factor input port. Next, it performs a complex multiplication on the two inputs. The complex result is added into a complex, floating point accumulation obtained from the RAM. The accumulation result is stored in the RAM, across the entire width of the RAM (36-bits for a two bank 512 X 18 RAM).

Two operating modes are required in the multiply/accumulator application, non-polarization and polarization.

In non-polarization mode, points to be multiplied enter the chip in pairs, one pair every second 32-MHz clock. On average, the RAM must be read, a multiplication done, the result added to the accumulator, and the sum stored back in RAM in the two available clock cycles.

The polarization mode requires that point pairs be multiply/accumulated, one pair every clock cycle. At least two consecutive (or two closely spaced in time) point pairs entering the chip are added into the same accumulation result between reading the accumulator partial sum and writing that sum back into the RAM. Thus, on average, the RAM must be read, two multiplications made, both results added via the accumulator to the same accumulation sum, and the new accumulation stored back in RAM, each two clock cycles.

In both of these modes, the RAM access requirement is one RAM read or one RAM write per clock cycle.

3.1.3.5 Number Controlled Oscillator

The ASIC has a 10-bit slice of a Number Controlled Oscillator (NCO) on

board for the Fringe Rotator and FSTC correction. The NCO is functionally independent of the rest of the device. Two secondary storage registers for loading an initial oscillator phase and phase increment are required. These secondary storage registers have serial I/O. A LOAD line strobes these values into the active registers. The NCO adder carry out line is pipelined so that any number of the bit slices may be tied together, to make a larger NCO. The accumulator is clock enabled, to allow selectable update rates. An intermediate storage register captures the NCO phase when instructed by the external STORE ACCUMULATOR line. The accumulator rewinds back to this stored phase, upon command of the external REWIND signal.

3.1.4 Fundamental Operating Modes for the VLBA1 ASIC

The ASIC has a 32 bit control word that is shifted in. The control word contains 9 bits that are used to define the fundamental operating modes for the chip. These are bits numbered B1, B3, B5, B6, B7, B15, B23, B25 and B26. (Section 3.1.6 below contains a detailed definition of each bit, B0 - B31.) The fundamental modes are defined as follows:

When reference is made to "upper" and "lower" pipelines, where the four pipelines on an FFT card are numbered 0 - 3:

For ASIC stages 0 - 4:
upper pipelines are 0 and 2, lower pipelines are 1 and 3

For ASIC stage 5:
upper pipelines are 0 and 1, lower pipelines are 2 and 3

MODE MNEMONIC	FULL NAME
R2LT2K	RADIX 2, LESS THAN 2K FFT SIZE
R2EQ2KUPR	RADIX 2, 2K FFT SIZE, UPPER PIPELINE
R2EQ2KLWR	RADIX 2, 2K FFT SIZE, LOWER PIPELINE
R4BYP	RADIX 4, BYPASS
R4	RADIX 4
R41K2KSTG4	RADIX 4, 1K or 2K FFT SIZE, STAGE 4
R41K2KSTG0UPR	RADIX 4, 1K or 2K FFT SIZE, STAGE 0, UPPER PIPELINE
R41K2KSTG0LWR	RADIX 4, 1K or 2K FFT SIZE, STAGE 0, LOWER PIPELINE
MACNP	MAC, NON-POLARIZATION
MACPA0	MAC, POLARIZATION, ARRAY 0
MACPA1	MAC, POLARIZATION, ARRAY 1
TESTMODEx	MISC TEST MODES

The table below defines the states of the nine bits for these fundamental modes.

		B3	B15	B26	B25	B1	B23	B7	B6	B5	<-- Bit nr	
		TEST	R4 CNT 0	FFT ST0	R4 1K 2K	RAM ON IN	EXT IN	NM/ AUX ---	2K/ <2K ---	R4/ R2 ---	Decimal equivalent of B7,B6,B5 (used in some software)	
M								UPR/	BP/	P/	where X is taken as 0	
O	M							LWR	NBP	NP		|
D	A								---		MODE	|
E	C								A1/0		MNEMONIC	|
--	--	----	----	----	----	----	----	----	----	----	----------	---
0	0	0	1	0	0	1	0	1	0	0	R2LT2K	4
1	0	0	1	0	0	1	0	X	1	0	R2EQ2KUPR	2
2	0	0	0	0	0	1	0	X	1	0	R2EQ2KLWR	2
3	0	0	1	0	0	1	0	1	1	1	R4BYP	7
4	0	0	1	0/1	0	1	0	1	0	1	R4	5
5	0	0	1	0	1	1	0	X	0	1	R41K2KSTG4	1
6	0	0	1	1	1	1	0	1	0	1	R41K2KSTG0UPR	5
7	0	0	1	1	1	1	0	0	0	1	R41K2KSTG0LWR	1
8	1	0	0	0	0	0	1	1	X	0	MACNP	4
9	1	0	0	0	0	0	1	1	0	1	MACPA0	5
10	1	0	0	0	0	0	1	X	1	1	MACPA1	3
11	0	1	0	0	0	1	1	1	0	1	TESTMODE1	5
12	0	0	1	0/1	0	1	1	1	0	1	TESTMODE2	5
13	0	0	1	0/1	0	1	1	1	0	1	TESTMODE3	5
(R4CNT0 = 0 : CNT0 = CT0								R4CNT0 = 1 : CNT0 = CT\0)				

There are other bits, such as EXTEXT (which is used in TESTMODE3) which are not shown above. See Section 3.1.6 for detailed description of each control bit.

A control bit, SEQ3REV, is used to specify when an ASIC address generator must invert the LSB if the address sequence is for stage 3. This is always asserted in the "LWR" ASICs at stages 3 and 4.

A control bit, TOGCEN8, is normally set = 0 in FFT modes, and set = 1 in MAC modes. This bit can be set = 0 in MACPA0 mode to allow accumulation of R*R and L*L in the same chip, for possible use by the Indian correlator.

A control bit, R2SUB, is used to cause a radix 2 stage to generate the "mirror image" point (A-C) in place of the "normal" point (A+C).

MAC is not a control bit, but is shown above for reference. MAC mode is selected when the input pins NCOOE and TWIDOE are both at logic 1.

Testmode1= test of address generators and ram

Testmode2= same as R4 except EXTIN=1 to bypass ram

Testmode3= same as R4 except force external write enables

3.1.5 ASIC Pin Functionality

This section describes the pin functionality of the NRAO VLBA Asic.

/corrdwgs/ga/doc/d001101.wk1 is the Lotus worksheet tabulating the asic pins. Part numbers ending in D or U indicate a pull-down or pull-up resistor for the pin.

The asic pins alphabetically by pin name, are described in Appendix IV.

/corrdwgs/ga/doc/d001103.prn lists the pins in pin numeric order.

/corrdwgs/ga/doc/d001104.prn, derived from d001104.wk1, lists the timing characteristics of each pin.

3.1.5.1 General Information

The two main asic modes are MAC and FFT. MAC mode is selected by having the pins NCOOE and TWIDOE both high. In MAC mode, the NCO is not used. Thus many of its pins are available for MAC functions.

3.1.5.2 Data Formats

MAC DATA OUTPUT FORMAT - 15,15,6 36 BIT NUMBERS

When AUXOUT is logic zero, the 18 bits selected to the ASIC output are:

```
d17  d16  d15  d14 d13 d12 d11 d10 d9 d8 d7 d6 d5 d4 d3 d2 d1 d0
---- ---- ---- --- --- --- --- --- -- -- -- -- -- -- -- -- -- --
exp5 exp4 exp3 i14 i13 i12 i11 i10 i9 i8 i7 i6 i5 i4 i3 i2 i1 i0
```

When AUXOUT is logic one, the 18 bits selected to the ASIC output are:

```
d17  d16  d15  d14 d13 d12 d11 d10 d9 d8 d7 d6 d5 d4 d3 d2 d1 d0
---- ---- ---- --- --- --- --- --- -- -- -- -- -- -- -- -- -- --
exp2 exp1 exp0 r14 r13 r12 r11 r10 r9 r8 r7 r6 r5 r4 r3 r2 r1 r0
```

Where the data is in 15,15,6 format (15 bit real 1's complement number, 15 bit imaginary 1's complement number, and a 6 bit 2's complement exponent).

TWIDDLE FACTOR INPUT FORMAT

INTERNAL TWID 554		STG0 +504	NCO OUT	STG1-5 550	MAC 444
	D0	T0	NCO0	T0	0
REAL	D1	T1	NCO1	T1	T1
SM #	D2	T2	NCO2	T2	T2
	D3	T3	NCO3	T3	T3
sign	D4	T4		T4	T4
	D5	0		T5	0
IMAG	D6	0		T6	T6
SM #	D7	0		T7	T7
	D8	0		T8	T8
sign	D9	T9		T9	T9
	D10	T5	NCO4		T5
EXP	D11	T6	NCO5		T0
MAG	D12	T7	NCO6		NCOSI
	D13	T8	NCO7		NCOCI

The table above explains the data multiplexing for the asic twiddle

factor.

Internal to the asic, the twiddle factor is represented by a 554 number. This is represented in the first column. D0 thru D5 are the real mantissa as a sign magnitude number. D5 thru D9 are the imaginary mantissa as a sign magnitude number. D10 thru D13 are a common exponent magnitude, with an implied negative sign.

The second column labeled STG0 +504 shows which input pins are used in FFT stage 0. T0 thru T9 represent the input pins TWID[0..9]. The control register bit FFTST0 being 1 says this is fft pipeline stage 0. Internal asic multiplexers allocate the bits as shown. On the card, T4 and T9 would be hard wired to 0. This gives the window input as a positive real number, with a 4 bit exponent.

The third column shows the corresponding NCO bits used to load the window ram. This occurs when TWIDOE=1 and NCOOE=0. Then the NCO output is directed out TWID0-3 and TWID5-8 as shown.

The fourth column, STG1-5 550, shows the format for FFT stages 1 thru 5. The control word bit FFTST0 would equal 0. A 550 number is derived from T0-T9 as shown. The exponent is internally set to 0.

The fifth column shows the 444 number used in MAC mode. The mantissas have 3 bit magnitudes and a sign. MAC mode is defined by NCOOE=1 and TWIDOE=1. The input pins NCOSI and NCOCI are used to furnish D12 and D13. The NCO is not used in MAC mode.

NDATAIN[0..17] or AUXIN[0..17] INPUT FORMAT
ALSO OUT[0..17] OUTPUT FORMAT IN FFT MODE

These are 774 numbers. There are seven bit, sign-magnitude real and imaginary mantissas. There is a 4 bit exponent magnitude, with an implied negative sign. The bits are allocated as shown below:

d17	d16	d15	d14	d13	d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0
exp3	exp2	exp1	exp0	i6	i5	i4	i3	i2	i1	i0	r6	r5	r4	r3	r2	r1	r0

3.1.5.3 Pin Descriptions

See Volume II for Chip Pinout Diagram

AUXIN[0..17]
774 Auxiliary input bus. This is used in FFT mode to bring in data from other pipelines for FFT sizes greater than 512.

CLK - PIN H13
Main logical clock for the ASIC. This does not affect the NCO

CLKA - PIN A07
This clock is nanded with the pin CLK to form the WE pulse for the onboard rams. This pin was provided in case the WE pulse width needed to be adjustable. Having this pin tied high gives a default timing which appears acceptable.

CRBIT - PIN N12
Control Register Bit. Based on the control word bit WEOUT, this outputs the control word bit CRBIT, or the ram WE\ pulse. CRBIT can be used to control external functions. CRBIT needs to be latched in via CRSTB. WE\ provides a means to monitor the write pulse timing.

CRSFTCLK - PIN L09
Control Register Shift Clock. This clock shifts in the asic control word, clocking in data from the pin CRSI.

CRSI - PIN N10
Control Register Serial Input. This data is clocked in by CRSFTCLK to shift in the 32 bit asic control word.

CRSO - PIN N11
Control Register Serial Output. This is the output of the shift register which receives the asic control word, where CRSI is the data and CRSFTCLK is the clock. This allows daisy chaining many asics together to form a card based control word.

CRSTB - PIN M10
Control Register Strobe. The control word in the asic is double buffered. This positive pulse latches the control word, once it has been shifted in.
Note:
NCORW*CRSTB=1 will cause the DECA internal flipflop to reset. The DECA flipflop causes the ram bank address generator to toggle states every 512 clocks. Thus extra CRSTB pulses could cause extraneous ram bank switching. The NCORW\ gating is for test purposes.

DATAENBL\ - PIN N07
Data Enable\. In FFT mode, this becomes the PULSARG\ (Pulsar Gate\). PULSARG\=1 causes OUT[0..17] to be 0. In MAC mode, this becomes CLRACCUM where a logic one clears the accumulator.

EXPOF\ - PIN M09
Exponent overflow\. This gives a logic 0 if the EXPOFENBL bit of the control register is set and there is an overflow when adding the exponents during the complex multiply. This output is open drain. Open drain allows many asics to have their outputs tied together. An external pull up resistor is required. If there are exponent overflows, the exponent can be adjusted via ADJEXP[0..1] in the control word.

EXTADD\[0..8]
External Address\[0..8]. These pins can either provide a 9 bit external ram address input, or the 9 msb's of the NCO output. The pin NCODOUT9 provides the ninth NCO output bit. The pins NCOOE and TWIDOE control the functionality of the pins. If NCOOE*TWIDOE\ equals 1, the pins provide the NCO output. See Section 3.3.1 for a detailed explanation of the bus function selection on the FFT card.

EXTASEL - PIN N08, EXTBSEL - PIN M08
These pins control the internal/external ram address generator selections as follows:

1) Internal chip signals EXTA and EXTB control the multiplexers that select the internal address generators or the external source of address.

   When EXTA or EXTB is zero, the corresponding internal address generator is selected. When EXTA or EXTB is one, the external address bus is selected.

2) EXTA and EXTB may be generated internally or may be controlled from two chip pins, EXTASEL and EXTBSEL. When the EXTEXT bit in the control word is 0, EXTA and EXTB are generated internally. When EXTEXT is 1, EXTA and EXTB are controlled from EXTASEL and EXTBSEL.

3) EXTA and EXTB are only generated internally when they need to toggle. For cases where they need to be static, control them by EXTASEL and EXTBSEL.

4) For FFT stage 0, writing to ram needs an external address, while reading from ram needs an internal address. For FFT stage 5 the opposite is true. Thus, in both of these cases, EXTA and EXTB need to toggle every FFT cycle. EXTEXT is set 0 in the control word and the EXTASEL and EXTBSEL pins have no effect. Thus, for FFT stages 0 and 5 EXTEXT=0.

5) For FFT stages 1-4, the internal address generators are always used, thus EXTA and EXTB are static and controlled from the EXTASEL and EXTBSEL pins so EXTEXT is set 1 in the control word. Thus, for FFT stages 1-4 EXTEXT=1.

6) For MAC mode, the EXTASEL pin is internally multiplexed to control both sides of the ram. The EXTBSEL pin becomes the COL\ function. For all MAC modes, writing to ram needs an internal address, while reading from ram needs an external address. Thus EXTA and EXTB need to toggle. EXTEXT is set to 0 in the control word which makes the EXTASEL pin have no effect. The EXTASEL pin having an effect was a fallback position which was not implemented.

7) For MAC mode, EXTBSEL becomes the COL\ function. COL\ + ROW\(WEIN\B) = DEN\0 which output enables the OUT[0..17] bus. Thus in the array of a MAC card, when a chip has both ROW\ and COL\ selected, it will output enable. All the asics OUT[0..17] are tri-stated together.

FCLK - PIN B07
Fast Clock. This function was deleted after the cards had been laid out. It is not connected internally.

NCOCE\ - PIN M04
Number Controlled Oscillator Count Enable\.
For MAC mode the pin becomes MACWE\, MAC Write Enable\.
NCOCE\ count enables the NCO. This allows varying the count rate.
MACWE\ in MAC mode strobes the 36 bit output into the asic output register. The 36 bits are output in 18 bit chunks selected by AUXOUT, when ROW\ and COL\ are selected.

NCOCI - PIN L04
NCO Carry In. TWID13 in MAC mode.
NCOCI allows cascading NCOs on multiple asics. Both NCOCI, the carry in, and NCOCO, the carry out, have pipeline stages. This causes a multiple asic, cascaded nco to have subsequent stage's outputs to be offset 2 bits in time.
TWID13 provides bit 13 of the twiddle factor in MAC mode.

NCOCLK - PIN J03
NCO Clock. Main clock for the NCO. Not used in MAC mode.

NCOCO - PIN M06
NCO Carry Out. The carry out used in cascading NCOs on multiple asics. NCOCO is connected to NCOCI of the previous asic. Both NCOCI, the carry in, and NCOCO, the carry out, have pipeline stages. This causes a multiple asic, cascaded nco to have subsequent stage's outputs to be offset 2 bits in time.

NCODOUT9 - PIN H02
The tenth NCO output pin. The other 9 pins are shared with the external address bus.

NCOLDIN\ - PIN N04
NCO Load In\
The data into the NCO is double buffered. This synchronously strobes the data into the NCO rate register and accumulator.

NCOOE - PIN M03
NCO Output Enable. If NCOOE*TWIDOE\ equals 1, the pins EXTADD\[0..8] provide the NCO output. See Section 3.3.1, for a detailed explanation of the function selection on the FFT card.

NCORW\ - PIN L06
NCO Rewind\. NCORW\ synchronously causes the NCO accumulator to rewind to the value stored by the last instance of NCOSTA\.
Note:
NCORW*CRSTB=1 will cause the DECA internal flipflop to reset. The DECA flipflop causes the ram bank address generator to toggle states every 512 clocks. Thus extra CRSTB pulses could cause extraneous ram bank switching. The NCORW\ gating is for test purposes.

NCOSI - PIN N02
NCO Serial In. TWID12 in MAC mode.
NCOSI provides the data to be shifted in by the NCOSRCLK. The data consists of a ten bit rate followed by a ten bit initial phase. Data is shifted in lsb first. NCOSO allows cascading NCOs. TWID12 provides bit 12 of the twiddle factor in MAC mode.

NCOSO - PIN M05
NCO Serial Out. NCOSO provides a serial out of the NCO data, to allow cascading the data loading of multiple NCOs.

NCOSRCLK - PIN L05
NCO Shift Register Clock. NCOSRCLK is the clock used to shift the data present at NCOSI into the NCO.

NCOSTA\ - PIN N05
NCO Store Accumulator. AUXOUT in MAC mode.
NCOSTA\ causes the synchronous storing of the NCO accumulator value. The stored value is rewound to on the NCORW\ signal.
AUXOUT controls the multiplexer to chose which half of the 36 bit number to output in MAC mode. AUXOUT = 1 selects the half with the real mantissa.

NDATAIN[0..17]
Normal Data In[0..17]. This provides the normal, as opposed to auxiliary data input, for a 774 number into the ASIC.

OUT[0..17]
This provides the output of a 774 number in FFT mode, or the output of half a 15,15,6 number in MAC mode. In MAC mode the outputs are tri-stated controlled by COL\ + ROW\(WEIN\B) = DEN\0 .

PNAND - PIN H11
PNAND provides the output of the PNAND tree used in chip testing.

R2CNT0 - PIN L02
R2CNT0 provides an output signal at half the frequency of CLK. The control word bit R2C0INV allows inverting the phase of R2CNT0. The phase is initially determined relative to RST\ into the chip. R2CNT0 is output from a flipflop clocked by the internal FCLK.

RAMTEST\ - PIN L10
RAMTEST\ places the internal rams in test mode for chip factory checkout.

RST\ - PIN J13
Reset\. RST\ resets the internal asic counter which generates control signals and internal ram addresses.

TWID[0..9]

Twiddle[0..9]. This provides the twiddle factor input in FFT mode. In MAC mode it provides the input one of the two numbers to be multiplied. If TWIDOE=1 and NCOOE=0, then the NCO output is fed out of the TWID bus as defined in the DATA FORMAT section. This provides the ability to load the window ram on the FFT card.

TWIDOE - PIN N03
Twiddle Output Enable. When If TWIDOE=1 and NCOOE=0, then the NCO output is fed out of the TWID bus as defined in the DATA FORMAT section. If TWIDOE=0, the FFT card twiddle factor is input. If TWIDOE=1 and NCOOE=1, then the card is in MAC mode, and the twiddle bus is used to input a number to be multiplied.

WEIN\A - PIN L08
In FFT mode, this affects Ram A. In MAC mode, this affects ram A and B.
Logic 0 allows internal ram write enable generation.
Logic 1 inhibits ram write enable generation.

WEIN\B - PIN N09
In FFT mode, this affects Ram B. In MAC mode, this becomes ROW\.
Logic 0 allows internal ram write enable generation.
Logic 1 inhibits ram write enable generation.
In MAC mode, ROW\=0 and COL\=0 selects the asic to activate the OUT[0..17] tri-state bus.

3.1.6 VLBA1 ASIC Control Word

3.1.6.1 Program Control Word Bit Format

The program control word to the ASIC is a 4 byte word that is shifted serially, bit 0 first, into the ASIC. A latch provides secondary storage. The layout of bits within the four bytes is as follows:

NM/ AUX	2K/ <2K	R4/ R2	TOG CEN8	TEST	PG ENBL	RAM ON IN	CARD USE	
UPR/ LWR	BP/ NBP A1/0	P/ NP						BYTE #0
B7	B6	B5	B4	B3	B2	B1	B0	
R4 CNT0	RD STG 2	RD STG 1	RD STG 0	ADJ EXP 1	ADJ EXP 0	DISC INT WE	EXT EXT	BYTE #1
B15	B14	B13	B12	B11	B10	B9	B8	
EXT IN	EXP OF ENBL	R2 SUB	R2C0 INV	NCO FRZ	WR STG 2	WR STG 1	WR STG 0	BYTE #2
B23	B22	B21	B20	B19	B18	B17	B16	

```
                RAM  WE   FFT  R4   SEQ3
                OFF  OUT  ST0  1K   REV          BYTE #3
                                2K
---- ---- ---- ---- ---- ---- ---- ----
B31  B30  B29  B28  B27  B26  B25  B24
```

3.1.6.2 Functional Description of Control Word Bits

The following pages describe the functions of the bits. When reference is made to "upper" and "lower" pipelines, where the four pipelines on an FFT card are numbered 0 thru 3:

For ASIC stages 0 thru 4:
upper pipelines are 0 and 2, lower pipelines are 1 and 3.
For ASIC stage 5:
upper pipelines are 0 and 1, lower pipelines are 2 and 3.

Bit 31
Bit 30
Bit 29: Unused bits

Bit 28:
RAMOFF 1= de-select rams (used only for IDD current test)

Bit 27:
WEOUT 1= switch an always enabled ram write pulse to be output from the CRBIT pin. Looking at the write pulse with an oscilloscope can be used in calibrating the CLKA delay line value used.

Bit 26:
FFTST0 1= chip is at stage 0 of fft pipeline
0= otherwise

This bit is involved with multiplexing of input pins, where the twiddle inputs are "positive 504" in stage 0 and are "550" in other fft stages. Additionally, this bit is used in conjunction with the R41K2K and UPR/LWR bits to determine the clock enables for, and alternation between the NORM and AUX inputs at stages 0 and 4.

Bit 25:
R41K2K 1= chip is either at stage 0 or stage 4 of fft pipeline and is involved in a 1K or 2K size fft
0= otherwise

Used in conjunction with FFTST0 and UPR/LWR as described above. When asserted in stage 0, pairs of points are simultaneously clocked into both the NORM and AUX inputs on every other clock. In a "upper" pipeline, the NORM input is written to ram in one clock cycle, followed by the AUX input in the next clock cycle. In a "lower" pipeline, the AUX input is written to ram first, followed by the NORM input. This allows the external address sequence to both chips to be identical, except for a one cycle delay in the address going to the "lower" pipeline.

When asserted in stage 4, either "upper" or "lower", four successive points are clocked into the NORM input and written to ram, followed by four points at the AUX input. This puts

the "even" butterflies in the "upper" pipeline, and the "odd" butterflies in the "lower" pipeline.

Bit 24:
SEQ3REV 1= chip is at stage 3 or stage 4 of a "lower" fft pipeline
0= otherwise

When asserted, the address generator that is counting in "stage 3" sequence will invert the ls bit of the address, having the effect of reversing the order of butterflies. This results in the lower pipeline doing butterflies involving odd numbered points while the upper pipeline is doing even numbered points and vice versa. This is only of interest in the case of 1K and 2K fft sizes, where it is necessary in order to have each pipeline handle evenly spaced spectral points at the FSTC portion of stage 5.

This bit is always asserted as described above, even when doing smaller ffts. This allows us to always use the same twiddles at stage 3. (Stage 3 lower always does butterflies in "reversed" order.)

Bit 23:
EXTIN 1= select the external data (NORM or AUX) as the input to the butterfly (normally only for MAC mode, possible for test purposes)
0= select the ram data as the input to the butterfly (normally for FFT modes)

Bit 22:
EXPOFENBL 1= enable the exponent overflow detection signal out of the chip
0= disable the output (default for MAC mode)

In normal use, all exponent overflow outputs are enabled and all six chips in a pipeline are tri-stated together with a pull-up. Each output is normally in the high impedance state, and an output only pulses low when overflow occurs. If overflow is occurring often enough, the outputs can be selectively enabled in order to determine which stage or stages are generating the overflow.

Bit 21:
R2SUB 0= normal

1= radix 2 "mirror" mode where the A-C result is kept rather than the A+C result (when set to 1 in radix 2 mode, the SUBA signal is forced to SUBTRACT instead of ADD)

Bit 20:
R2C0INV Controls the phase of the R2CNT0 output

Bit 19:
NCOFRZ 1= allow the NCO accumulator to be loaded with data in the normal manner, but "freeze" the NCO operation so that the value will not ramp up. This is required when using the NCO for loading rams.

0= allow normal NCO operation.

Bit 18 (ms)
Bit 17
Bit 16 (ls):
WRSTG 0 thru 4 = sequence 0 thru 4 (set = 4 for MAC mode)
5 = not used
6 = external address sequence at stage 0
7 = not used

This three bit field defines the address generation sequence to be used for writing to ram.

Bit 15
R4CNT0 1= CNT0 same phase as CT\0
0= CNT0 same phase as CT0
This bit goes to an exclusive NOR that generates the 16 MHz CNT0 signal as a function of the lsb of the control logic counter. R4CNT0 is low for all MAC modes and for radix 2, 2K in a "lower" pipeline. It is high for all other FFT modes.

When stage 5 is doing radix 2 2K, the "lower" chip selects an internal one clock cycle delay for the FFTINIT pulse, because there is a one cycle skew between the input data for the upper and lower chips. The R4CNT0 control bit compensates for this skew so that the butterfly operations are not skewed. Thus the internal address generators are skewed for purposes of writing to ram, but the butterflies are not skewed so that a common external address generator may be used.

When stage 0 is doing 1K or 2K ffts, the pairs of input points are also skewed between upper and lower. In this case, the internal address generators are used for reading the ram (unlike stage 5 where they are used for writing the ram). Thus the internal address generators may not be skewed, and the required skew for writing to ram must be handled by the control logic and by separate external address sources.

In summary, at stage 5, the skew for writing to ram is provided by an internal FFTINIT delay which is compensated for by R4CNT0. At stage 0, other means are required. The difference is based on whether the internal address generators are used for reading or writing.

Bit 14 (ms)
Bit 13
Bit 12 (ls)
RDSTG 0 thru 4 = sequence 0 thru 4 (set = 4 for MAC mode)
5 = not used
6 = not used
7 = external address sequence at stage 5

This three bit field defines the address generation sequence to be used for reading from ram.

Bit 11 (ms) (ignored in MAC mode, use 0 for default)
Bit 10 (ls)
ADJEXP 0 thru 3 = the number to "add" to the exponent at the input to the butterfly. (The exponent is an implied negative number that is tending to grow more negative. The value added to the exponent "pulls" the result towards zero.) This adjustment will be a function of the type of observation. This field is ignored in stage 0.

In the TMS32020 simulator, the defaults for this parameter were to add one to the exponent at stages 0 thru 2 and to add two at stages 3 and 4 when simulating continuum or wide spectral lines. When simulating narrow spectral lines, we added one at stages 0 thru 4. Stage five either had zero or one added (I believe zero is appropriate). (This field was not ignored in stage 0 of the simulator.)

Bit 9
DISCINTWE 1= inhibit the internal ram write enables
0= enable the internal ram write enables

Normally this bit is zero. This function is provided as a "safety net" only. The external write enable bar pins are normally zero, other wise they would inhibit the internal write enables.

Bit 8
EXTEXT 1= force control of the ram address multiplexers to come from chip input pins rather than from internal control logic
0= allow internal control of the multiplexers

At fft stages 1 thru 4, no external address is used, so the multiplexers need to always select the internal address generators. Thus EXTEXT is set to one, and the chip EXTA and EXTB input pins are tied low to provide the static control line to the multiplexers.

At fft stages 0 and 5 and in MAC modes, the multiplexers need to alternately select both internal and external address sources. For these modes, EXTEXT is set to zero, allowing the control logic to toggle the multiplexes as required.

As a safety net, EXTEXT could be set high, and the control could be generated externally.

NOTE:
For both safety nets of external write enables and external ram Multiplexer control, there are two chip pins provided (WEIN\A, WEIN\B, EXTA and EXTB). These pins are shared with other functions such that in MAC modes there is only a single WEIN\ and a single EXT pin. This is because in MAC modes, the ram is a single 36 bit wide ram, where "bank switching" is a logical concept controlled by the MS bit of the address, whereas in FFT modes, the ram is physically two separate ram banks, each needing a separate write enable and address source.

Bit 7
NM/AUX
UPR/LWR This is a dual function bit. In certain modes, this bit is set high, to select the normal data input. (If set low, it would select the auxiliary data input. Normally, the aux input is never statically selected.) Set = 1 for MACNP and MACPA0, = 0 for MACPA1.

In some modes, the input toggles back and forth between norm and aux. For these modes this bit is a don't care, except for radix 4, 1k or 2k modes in stage 0, where this bit is used to specify the upper or lower pipeline. 1= upper, 0=lower.

Bit 6
2K/<2K
BP/NBP
A1/0 This bit is a triple function bit.

For radix 2 modes: 1= 2K fft size
0= less than 2K fft size.

For radix 4 modes: 1= bypass
0= not bypass

In MAC polarization mode: 1= array 1
0= array 0

Bit 5
R4/R2
P/NP This bit is a dual function bit.

For fft modes: 1= radix 4
0= radix 2

For MAC modes: 1= polarization
0= non-polarization

Bit 4
TOGCEN8 0= FFT 1= Normal MAC modes
Can be set = 0 in MACPA0 mode to allow the Right * Right and Left * Left mode for possible use by the Indian project.

Bit 3
TEST 1= test mode
0= not test mode

This bit is used for certain address generator and ram tests. At present, when test is set high, it forces FFTONOUT low to allow ram bank B to be routed to the chip output. It also enables the dual function chip pin NCOSTA\ to be routed to the AUXOUT select line so that either the normal output or the aux output can be routed to the chip output. In both of these cases, the normal control logic would inhibit these routings unless the chip was in MAC mode.

Bit 2
PGENBL For FFT ASIC usage:
1= enable the use of the DATAENBL\ pin for use as the PULSARENA\ (Pulsar enable \ or pulsar gate)
0= disable the pulsar gate logic (ignore pulsar gate)

Due to subarrays, we need to be able to selectively enable chips to blank results in response to a pulsar gate signal.

For MAC ASIC usage:
1 = enable the use of the DATAENBL\ pin for use as CLRACCUM.
This should always be 1 for MAC mode, since CLRACCUM must always be enabled.

Bit 1
RAMONIN 1= route input data to ram
0= route MAC accumulator results to ram

Bit 0
CRBIT This bit is not used on the chip. CRBIT is output from the chip for control use on the FFT card.

3.1.6.3 Description of ASIC Control Words on the FFT Card

On the FFT Card, the control word shifts thru the ASIC's in the following order. The stages are 0 to 5, left to right. The pipelines are A, B, C, D, where A is pipeline 0, the pipeline at the top of the FFT Card.

Stage 0A, 1A, 2A, 3A, 4A, 5A, 5B, 4B, 3B, 2B, 1B, 0B, 0C, 1C, 2C, 3C, 4C, 5C, 5D, 4D, 3D, 2D, 1D, 0D. An independent clock, CRSFTCLK, shifts the control word. The control word enters 0A at CRSI\ and leaves 0D from CRSO\.

When loading RAM's (LDRAM to RAM's =1):
Set NCOFRZ = 1 in all ASIC's to stop the NCO clocks.

Stg3A provides Additional RAM inputs via RAMLD
(INIT\1,2, FSCE\, ST\0, ST\1)

In normal operation:
Set NCOFRZ = 0 in all ASIC's to allow them to count.

3.1.6.4 Control Bits Used for FFT Card Functions

CRBIT bits from certain ASIC's provide card level control functions as follows:

ADR0	Stg0D Input MUX control
ADR1	Stg1D Input MUX control
RADIX0	Stg3C Fringe Rotator Store Select Counter Radix
RADIX1	Stg2C Fringe Rotator Store Select Counter Radix
RADIX2	Stg1C Fringe Rotator Store Select Counter Radix
RADIX3	Stg0C Fringe Rotator Store Select Counter Radix
RADIX4	Stg0B Fringe Rotator Store Select Counter Radix
RADIX5	Stg1B Fringe Rotator Store Select Counter Radix
RADIX6	Stg2B Fringe Rotator Store Select Counter Radix
RADIX7	Stg3B Fringe Rotator Store Select Counter Radix
FSCONLY\	Stg5A FSTC only inverted.
SIZE\1	Stg4D Stage 5 angle mux control
OVSAMP\	Stg2D Oversampled\, tell if not oversampling.

Stage 0A has the WEOUT bit set in the control word. This causes an always enabled ram write pulse, WE\, to be output from CRBIT to a probe test point on the card. (WE\ = CLKA * CLK)

3.2 FFT Card General Description

3.2.1 Introduction

Refer to the k034d01.sch, the D sized, FFT Card Detailed Block Diagram. An abbreviated, B sized version is k032d01.sch. These are in Volume II.

The heart of the FFT card is the VLBA1 ASIC. The ASIC performs a radix 4 or radix 2 FFT butterfly. The ASIC also contains a 10 bit, bit-slice, Number Controlled Oscillator (NCO). The ASICs are arranged into four FFT engines. Each engine is six ASICs long, to allow a maximum FFT size of 2048 points (5 radix 4 FFTs and 1 radix 2 FFT).

The data enters the FFT Card, multiplexed from up to four PBIs. The data is fringe rotated, using ROMs and the NCOs. The data then goes into the

FFT engines. The FFT Control Card (FCC) provides the trig tables and control signals. By modifying the trig tables to the last stage of an FFT Engine, a Fractional Sample Time Correction (FSTC) is applied. The FSTC correction is generated using ROMs and the NCOs. Tape Validity and Pulsar Gate signals can be used to zero the data. Outputs from pairs of FFT pipelines are multiplexed together and sent on to the Multiply Accumulate (MAC) cards.

3.2.2 FFT Card Files

Schematics
/corrdwgs/fft/sch/SCCS/s.l002d39.sch is the top level schematic.
Most schematics are in Volume II.

/magnolia2/vlbsoft/pal_prom/pals/fft/SCCS contains the PAL files. They are listed along with their functional block diagrams in Volume II. An exception is the PAL7F listing in Appendix II. This list was too long to fit in the schematic.

/home/magnolia2/vlbsoft/pal_prom/proms/fftfstc/SCCS contains the fstc proms
/home/magnolia2/vlbsoft/pal_prom/proms/fftfrng/SCCS contains the fringe rotator proms

3.2.3 Control Words

The asic control words are serially shifted into the 24 asics. Each asic outputs from its CRBIT pin one bit from its 32 bit control word. Thus the card control word is shifted in as part of the asic control words. The control words shift in on the card edge pins CRSI\ for data and CRSFTCLK for clock. CRSO\ is provided as a serial output for checking the control word path.

3.2.4 Input Data

The 2 bit, differential ECL input busses are multiplexed as shown in /corrdwgs/rackwire/l025/l025d26.sch, The Famous Star Diagram, included in Volume II. Each pipeline gets to chose between 4, 2 bit input busses, which correspond to discrete Playback Interfaces (PBI). All four pipelines are controlled by a common 2 bit address ADR[0..1]. ADR[0..1] comes from the 24 bit card control word. ADR\[0..1] is output to the card edge, for use in the test fixture, but not in the system.

There is a pipeline register after each data bus is converted to TTL. There are second pipeline registers after the multiplexer.

3.2.5 Fringe Rotator

The following description describes one pipeline, since all are similar.

The 2 bit bus selected by the input multiplexer, next goes through the fringe rotator. The 2 bit data bus gets multiplied times a 9 bit NCO angle to give a 16 bit 772 output. The 772 number contains real and imaginary 7 bit sign-magnitude mantissas and a 2 bit exponent with an implied negative sign.

The 9 bit NCO angle is the 9 msb's of a 40 bit NCO angle, which consists of 4 cascaded ASIC's NCOs. The 40 NCO bits are assigned as follows: Stage 1-bits 39-30, Stage 0-bits 29-20, Stage 2-bits 19-10, Stage 3-bits 9-0. Stage 1 is used as the most significant asic since stage 0 uses the common external address/NCO output bus for an external address.

Just because there is an NCO byte in each stage, it is not necessarily associated with that stage. We only look at the msbyte of a 4 byte NCO for fringe and a 2 byte NCO for fstc. The fringe msbyte has to sync up with stage 0. The fstc msbyte has to sync up with stage 5.

The NCO rates and initial phases are down loaded from the FCC using the card edge pins FRSI for the data and FRSRCLK for the shift register clock. All 4 pipelines are connected in series for the purpose of shifting in this data.

The 4 cascaded NCOs act as a 40 bit adder. However, the carry-ins and carry-outs are pipelined in each asic. Thus in loading the shifted in data into the NCO's, the FFT card takes the card edge signal FRLOAD\ and delays it 2 clock pulses for each subsequent stage. Thus the outputs of each stage are time offset 2 clocks from the previous stage. Since the fringe rotator only looks at the output of the ms stage, this does not present a problem.

The fringe rotator is essentially in front of stage 0. It does not care at all what the timing of fftinit to stage 3 is, even though part of the NCO is from the stage 3 ASIC. We just need the NCO msbs output to line up correctly with the stage 0 input.

On the fringe NCOs:

Notice that FRLOAD\0 goes to the ls 10 bit byte in stage 3.
FRLOAD\1 goes to the next msbyte in stage 2.
FRLOAD\2 goes to the next msbyte in stage 0.
FRLOAD\3 goes to the msbyte in stage 1.
The stage 1 output is the msbyte and goes to the fringe rotator.

The timing is:

```
         FRLOAD\ -_-
         FRLOAD\0 -_-
         FRLOAD\1 ---_-
         FRLOAD\2 -----_-
         FRLOAD\3 -------_-
NCO REG X                012
NCO ACCUMULATOR OUT       012
FR OUT=DATA TO ASIC        012
FFTINIT\' TO STG0  -------_--
FFTINIT\ TO FFT CARD  ---_--
```

The FCC determines the timing of when it sends FRLOAD\.
FRLOAD\3 is the one we look at to see timing in relation to the system.

The actual multiplication is performed via two 2k x 8 registered PROM's. The prom data is stored in /vlbsoft/pal_prom/proms/fftfrng/SCCS/newu107.ls and newu108.ms.

The PROM output goes to the NDATAIN input of its pipeline's asic and the AUXIN input of a neighboring pipeline. The AUXIN input is used in 1k and 2k mode.

The current state of the NCO can be stored and rewound to. See Section 3.3.2.3 below for a detailed description.

3.2.6 Fractional Sample Time Correction (FSTC)

Refer back to section 2.2.3 for the FSTC generation from the FCC's point of view.

The FSTC corrects for the fractional portion of the delay, where the PBI

removed the integral number of bits delay. The FSTC is implemented by multiplying the FFT output by a rotating vector, e**(-jb). The angle b comes from the FSTC NCO.

The stage 4 and 5 asics are used in implementing the 20 bit FSTC NCOs. Refer to k034d01.sch in Volume II, the FFT Card Detailed Block Diagram. The radix 2 butterfly in the lower left shows how a is the stage 5 trig table angle and b the FSTC NCO output angle. Multiplying the butterfly input A * e**-jb, and the second butterfly input B * e**-j(a+b), gives the butterfly output (A + B*e**-ja) * e**-jb.

The block diagram in the lower right shows how this is implemented logically. The NCO angle b goes to both the small and big rom. The small rom outputs e**-jb. The big rom also gets the trig table angle a, and outputs e**-j(a+b). The two angles are input into the asic in sequential clock cycles. Thus each angle can change at 16 MHz. There is a pipeline register out of the big rom, clocked by a 16 MHz R2CNT0 signal from an asic. A registered multiplexer combines the two angles to enter into the asic.

For FFT sizes that do not use the radix 2 stage (1K, 256, and 64), it is required to multiply the first data point in times e**-jb, and zero the second data point. The FSCONLY\ bit from an asic CRBIT, clears the pipeline register on the output of the big rom. This zeros the second point.

The NCO angle needs to be constant for the two clock periods. This is implemented via a 16 MHz clock enable to the NCOs, FSCE\, which comes from the rams. See 1002d38.sch for related timing.

When doing FFT sizes less than 512, multiple FFTs are done in each 512 block. All the spectral point zeros are grouped together, then all the point ones, etc. The FSTC NCO stays constant until the point number changes. This is implemented via the clock enable FSCE\ as follows:

FFT Size (# points)	2K	1K	512	256	128	64
FSCE\ Rate(MHz)	16	16	16	8	4	2

The stage 5 trig tables are manipulated by the Angle Mux. See the block diagram, 1002d07.sch, and the actual schematic, 1002d06.sch. The trig table is only used when the radix 2 is needed. That is for 128, 512, or 2K FFTs. Otherwise, the FSCONLY\ control bit zeros the output from the big rom, which is derived from the trig table. For the radix 2 butterfly, the trig table changes at half the data rate. That is 16 MHz. Thus 2 trig tables can be time multiplexed in on the 32 MHz trig table input from the FCC. The 128/512 trig table input contains the 128 point and 512 point FFT trig tables multiplexed together. These signals are demultiplexed by going through 2 layers of registers, clock enabled by the 16 MHz R2CNT0 signals from different asics, as shown on the block diagram. The ASIC control bit R2C0INV controls the phase of the respective R2CNT0 signals, as shown on the table on 1002d07.sch in Volume II. The output of the 4 registers go to the 4 pipelines, with all 4 pipelines getting identical angles. These output registers are tristated with the angles for 2K FFTs. The CRBIT from asic 4D furnish the signal size\1 which controls the selection.

For 2K, stage 5 angles, each pipeline gets different angles. The angles are multiplexed into the card on the 32 MHz bus, 2K EVEN/ODD IN. A 2K FFT is accomplished in 512 clock cycles by having all 4 pipelines involved. This was necessary since the ASIC on-board rams only have 512 locations. The 1024 output points are divided among the pipelines such that points 0-510 even are output from pipeline 0, points 1-511 odd are output from pipeline 1, points 512-1022 even are output from pipeline 2, and points 513-1023 odd are output from pipeline 3.

The stage 5 angles were originally designed to be sent as a sine and

cosine from the FCC, on an 8 or 9 bit bus. It was later redesigned as a five bit index or angle. The sine and cosine were not necessary, since they are obtained from the roms on the FFT card. The msb of the five bit index represents ninety degrees. On the 9 bit bus, the four lsbs of the index are sent twice in two four bit sections, with a single msb. The FSTC rom contents generation was described in section 2.2.8.

The 2K EVEN/ODD bus contains the angles for points 0-511. These angles are between 0 and 90 degrees. Hence, the msb of the 5 bit index is zero. The index is sent twice as two, four bit sections of an eight bit bus. The msb is not sent since it is zero. As shown on 1002d07.sch, registers W, X and Y demultiplex the odd and even points to go to pipelines 0 and 1. The angles for points 512-1023 (going to pipelines 2 and 3) are obtained by adding 90 degrees to the angles for points 0-511. This is implemented by forcing the msb of the five bit index to be one. The even points are output from pipeline 2 and the odd points from pipeline 3. Size\1=0 selects the 2K angles on the data busses to the pipelines, as opposed to the 512/128 point angles.

Data is loaded into the FSTC NCOs via four serial data inputs (FSSIA, FSSIB, FSSIC, FSSID) and a common clock FSSRCLK. The signal FSCLOAD\ loads the shifted in rate and initial phase into the active registers. These signals are from the FFT Control Card. In the fringe rotator NCO, the load pulse to the more significant stage is delayed two clock periods from the load pulse from the previous stage. This is due to the pipelining of the NCO carry in and carry out. However in FSTC, the second load pulse is only delayed one clock period. This works because the clock enable causes the NCO to run at a 16 MHz or less. This allows time for the carry-in pipelining to propagate through at 32 MHz. This closer spacing of load pulses becomes essential when the NCO is running at 8 MHz. See corrdwgs/fft/sch/SCCS/1002d38.sch in Volume II.

FSST\, FSTC store, and FSRW\, FSTC rewind, are wired to the card edge. They are pulled high, and, in the system, the card pins are not wired.

3.2.7 FFT Card Rams

See 1002d05.sch in Volume II, for the schematic of the 5, 2k x 8 rams on the FFT card.

U123 provides the window which takes the place of the twiddle factor for stage 0. It is a 504, positive real number.

The rams provide the external addresses for stage 0 upper, stage 0 lower, and stage 5. These are the addresses for the internal asic rams.

Also provided are init\1 and init\2. These are the initialization pulses for the ASICs of stage 1 and 2. The initialization pulses for stage 0, 3, 4, and 5 come in, from the FCC, on card edge pins. ASIC initializations are spaced 11 clock cycles apart. Init\1 and init\2 coming from the ram was arbitrary and just meant to save card pins.

The Rams provide FSCE\, the FSTC clock enable, as was described above.

ST\0 and ST\1 provide the store pulses for the fringe rotator NCOs. See Section 3.3.2.3 below, for a detailed description of their use.

The card edge signal UPRMEM provides the option of storing a second set of ram values. In the system, UPRMEM is grounded at the card edge connector.

The rams are loaded by shifting serial data into the fringe rotator NCOs, then writing it into a ram location. The ram addresses are generated by the pals U126 and U127. These pals are described in Volume II. The pals use

the Palasm files ctrlsb.pds and ctrmsb.pds. A block diagram of the pals is presented in Volume II, on 1002d34.sch. U126 also contains a multiplexer used in the fringe rotator store pulse generation. The card edge signal, LDRAM, tells the pals to go into ram loading mode. In the normal operation mode, the pals count every clock cycle. In ram loading, when LDRAM=1, the card edge signal RAMCLK causes the rams to be written and the counter to increment. See the timing diagram on 1002d05.sch. The counter is reset by RST\ which is a delayed version of the card edge signal FFTINIT\. FFTINIT\ also generates the reset to stage 0. The details of setting up the asics so the NCOs will output the data are discussed in Section 3.3.1 below. Notice the NCO outputs via the twiddle factor port for the window data ram storage.

3.2.8 Output Multiplexer

See the right side of 1002d02.sch of the FFT Card Schematic, in Volume II. Each pipeline outputs data at 16 MHz, since the half of the spectrum with redundant information is thrown away. The outputs of pipelines 0 and 1 are multiplexed together to give a 32 MHz data output stream. The outputs of pipelines 2 and 3 are similarly multiplexed together to give another 32 MHz data output stream. An asic R2CNT0 provides the 16 MHz multiplexer word select. The outputs are converted to differential ECL for sending to the MAC cards.

3.2.9 Pulsar Gate

See 1002d04.sch of the FFT Card Schematic, in Volume II. The card edge signals PULSAR[0..3] are captured in the register U20, then go to the four stage 5 asics (U119, U99, U78, and U53). If the asic PGENABL, pulsar gate enable, bits in the stage 5 asic control words are set, then a logic 1 on the PULSARx line will zero the output of the asic. The pulsar gate is on the output of stage 5, to allow it to be implemented after the FFT is completed.

3.2.10 Data Invalids

The card edge signals DINVALID[0..3] are captured in register U113, at the bottom of 1002d02.sch of the FFT Schematic, in Volume II. The DINVALID' signals go to 1002d04.sch, to the four, stage 4 asics (U113, U91, U70, and U47). If the asic PGENABL (pulsar gate enable) bits in the stage 4 asic control words are set, then a logic 1 on the DINVALIDx line will zero the output of the asic. Since DINVALIDx is applied to stage 4, there is the stage 5 delay before any affect is seen on the output. There are programmable delays in the MCC for the number of fft cycle delays and the position in the fft cycle. It is acceptable to have the invalid applied in the middle of the fft chain, since the duration of the invalid spans the entire fft 16 usec time frame. Contrast this to the pulsar gate, which needed to be applied on individual spectral points. The pulsar gate is applied to the output of Stage 5 asics, after the fft.

3.2.11 Exponent Overflow Detection

The exponent at the output of each stage tends to "grow" more negative. The exponent adjust feature can be used to compensate for this, but can cause an exponent overflow. Exponent adjust is specified via the asic control word. If exponent adjust causes an overflow, the ASIC EXPOF\ pin goes low. This pin is open drain. Each pipeline's six outputs are tied together. There is a common pull-up resistor. The signals are captured in register U12, at the bottom left of sheet 1002d02.sch of the FFT Schematic. The signals leave the FFT Card as EXPOF\'[0..3]. Currently, the system does not monitor these bits, though they

can be probed on the backplane. The FFT Test Fixture does display the EXPOF\ bits on the front panel LEDs.

3.2.12 Card Clock Distribution

See the center of 1002d02.sch of the FFT Schematic. The 32 MHz differential ECL clock into the FFT card, is transformed into TTL in the 10125s, U45 and U33. These signals are known as FCLKx. FCLKx going to the ACTQ245 drivers (U68, U89, and U14) give the signals CK32x, which provides clocks to the asics and other functions.

Note the 245s need pin 1 pulled high through a resistor, rather than the pin being plugged into the socket. The ACTQ245 gave better clock stability than the 541 in the original design.

A clock from the 10125, U45, goes to delay line U67, and then to ACTQ245s (U68 and U14). The outputs of these ACTQ245s are called RCLKx. These go to any register whose output feeds an asic. This allows for the large data hold time spec of the asics. A delay of 5 ns was found to give good results.

A clock from the 10125, U45, goes to the delay line U88, and then to the ACTQ245, U89. These clocks (CA32A, CA32B, CA32G, and CA32D) go to the asics for shaping the write pulses to the internal asic rams. It was found that tying the output of U88 high gave a default write pulse shape that was acceptable.

3.2.13 ASIC Ram Address Routing

Refer to 1002d35.sch towards the beginning of Volume II. This shows the address generation for the asics of a pipeline. The top and bottom of the sheet show before and after an asic bank switch. Notice the same ram address sequence needs to be used for the internal ram being read by an asic and the ram being written to in the next asic down the pipeline. The initial stage 0 read uses an external address bus from the card rams to provide for scrambling the time ordered points. An external bus from the card rams, provides for output ordering for stage 5. Section 3.1.5 above gave more details on the asic functionality and address control.

3.2.14 1K/2K FFT Signal Routing

See 1002d36.sch in Volume II, for a diagram of the signal routing for 1k and 2k mode. All 4 pipelines are used to perform a 2k fft in 512 clock cycles.

An initial constraint is all points into the fringe rotators have to be evenly spaced in time.

The stage 0 input criss-cross provides the correct points needed for the data scrambling. Then, stages 0 through 3 do four 512 point ffts.

In an fft, every output point has inputs from every input point. Thus it is necessary to reunite the points. The criss-crosses from pipelines 0 to 1, and 2 to 3, allow the 1k fft outputs to come from stage 4.

A similar criss-cross allows the combining of points into stage 5 to give the 2k fft. Pipelines 0 and 2, and 1 and 3 are combined. This collects points from both 1k ffts. The drawing in the lower left shows how the first clock combines points 0 from pipeline 0, and 1024 from pipeline 2, into the pipeline 0 stage 5. The second clock combines point 512 from pipeline 0, and 1536 from pipeline 2, into pipeline 2 stage 5.

The following asic control bits have to do with setting up for 1k or 2k mode. See Section 3.1.6.2 above, for more details.

```
Bit 25
R41K2K    1=   chip is either at stage 0 or stage 4 of fft
               pipeline and  is involved in a 1K or 2K size fft
          0=   otherwise
Bit 6
2K/<2K
BP/NBP
A1/0      This bit is a triple function bit.
          For radix 2 modes:            1= 2K fft size
                                        0= less than 2K fft size.

          For radix 4 modes:            1= bypass
                                        0= not bypass

          In MAC polarization mode:     1= array 1
                                        0= array 0
Bit 5
R4/R2
P/NP      This bit is a dual function bit.
          For fft modes: 1= radix 4
                         0= radix 2

For MAC modes: 1= polarization
               0= non-polarization
```

3.3 NCO Bus Contention and Operating Modes

3.3.1 Tri-State Output Bus Contention

This section describes a means to prevent potentially catastrophic bus contention with the ASICs.

Two asic pins are required, labeled NCOOE and TWIDOE. The modes are assigned as follows:

			1=ENABLE OUTPUT		
MODE	NCOOE	TWIDOE	DOUT BUS	NCO/EXTADD BUS	TWIDDLE BUS
MAC	1	1	ROW*COL	0	0
(For MAC mode, no NCO or TWIDDLE outputs.)					
FFT0	0	0	1	0	0
(For FFT mode, no NCO or TWIDDLE outputs.)					
FFT1	0	LDRAM''	1	0	LDRAM''
(For loading ram window data from the twiddle output.)					
FFT2	LDRAM''	0	1	LDRAM''	0
(For loading ram data from the NCO output)					
FFT3	1	0	1	1	0
(For normal NCO angle output)					

The NCOOE and TWIDOE pins have internal pullups, so they default to logic 1 when floating.

The 1, 1 state of NOCOE, TWIDOE defines MAC mode, instead of using a bit from the control word. This is hardwired, so the bits will not toggle during

control word shifting. In MAC\ (FFT mode), the DOUT bus will be permanently enabled. In MAC mode, it will depend on the shared pins, ROW\ and COL\. Inside the ASIC, there are not be any registers associated with ROW\, COL\, or AUXOUT. This allows for bus contention protection on power on, rather than needing to wait for a clock to a register. The pins ROW\ and COL\ can be used for different functions in FFT mode, since MAC mode vs. FFT mode is hardwired.

In FFT mode, the bus contentions can come from the bidirectional busses used to load the ram. The LDRAM'' signal disables the ram outputs. Note LDRAM'' is a low speed signal. By using it to enable the ASIC, there can be no long term contention problem with the RAM. ASIC to ASIC contention is controlled by only wiring LDRAM'' to one ASIC of a set of several ASICs in parallel. Example: Four ASICs input the window bus in parallel. Only one asic will output to the bus during ram loading.

For the FFT card, hard wire the modes, using NCOOE and TWIDOE, as follows:

ROW	STG0	STG1	STG2	STG3	STG4	STG5
A	FFT1	FFT3	FFT2	FFT2	FFT3	FFT0
B	FFT2	FFT3	FFT0	FFT0	FFT3	FFT0
C	FFT2	FFT3	FFT0	FFT0	FFT3	FFT0
D	FFT0	FFT3	FFT0	FFT0	FFT3	FFT0

For STG0A, FFT1 provides for loading the window ram.
(The A in STG0A refers to the row, representing pipeline 0)
For STG0B, FFT2 loads the lower stage 0 external address.
For STG0C, FFT2 loads the upper stage 0 external address.
For STG0D, FFT0 prevents bus contention.

For all stage 1 and 4 outputs, FFT3 has the NCO permanently output the angles.

In stage 2, 3, and 5, FFT0 specifies no NCO outputs.

For STG2A, FFT2 provides for loading the miscellaneous ram outputs: INIT\1, INIT\2, FSCE\, ST\0, and ST\1.

For STG3A, FFT2 provides for loading the stage 5 external addresses. Stage 3A and the ram feed the input of a register. The register output feeds the 4, Stage 5 External Address inputs in parallel. This arrangement provides adequate fanout and short paths.

Also, notice all ram loading is done via the fringe rotator NCO's (Stages 0 through 3). When loading the rams, have the ASIC control bit NCOFRZ = 1 to prevent the NCO's from counting.

To test the tristate outputs for leakage, it is necessary to put them all in the hi-Z state. This can be accomplished by having NCOOE TWIDOE as 1 1, and either ROW\ or COL\ as a 1.

The momentary contention on switching LDRAM'' will not be worried about, since this only switches every few hours.

3.3.2 NCO Operating Modes

3.3.2.1 Introduction - Fringe Cycles

Four NCOs are tied together and are supplied with 40-bit initial phases and 40-bit fringe rates every fringe cycle. At the start of a fringe cycle the two coefficients shifted in during the last fringe cycle are clocked into the active

registers.

3.3.2.2 Normal Tracking of Fringe Phase (Non-Overlapped FFTs)

At the end of every FFT cycle, at point 512, the fringe phase and pipeline carries are stored into the capture register and rewound after the inter FFT cycle pause (a.k.a. the four cycle gap, since it is four, 32MHz clock cycles long). Except for this brief capture and rewind, the NCO runs continuously, independent of FFT size.

EXAMPLE:

FFT CYCLE	FIRST DATA POINT PRESENTED TO FRINGE ROTATOR	STORE DATA PT
0 -------->	0	512
1 -------->	512	512
2 -------->	1024	512
3 -------->	1536	512

3.3.2.3 Fringe Tracking with Overlapped FFTs

Overlapping involves backing up and reusing a portion of the data from the previous fft.

When overlapped FFTs are being performed, the time order of the bits presented to the fringe rotator will not be a simple monotonic function. Two parameters are involved when overlapping:

1) The FFT size (F)
2) The overlap factor (V)

If the FFT size is less than 512, this adds a complication. The data is still handled in 512 point chunks. Multiple smaller ffts are packed into the 512 points. This necessitates a minor cycle to advance within the small fft, and a major cycle to jump to a new block of 512 points.

EXAMPLE: 128 Point FFT, Overlap factor of 4

The first bit presented to the fringe rotator in FFT cycles subsequent to the start of a fringe cycle can be tabulated from these two parameters:

F=128, V=4, F/V=32

FFT CYCLE			FIRST POINT		STORE DATA PT
0	-->		0	0	32
1	-->		0 + F/V	0+32	32
2	->		0 + 2F/V	0+64	32
3	--->		0 + 3F/V	0+96	416
4	|		0 + 4F/V	Only 4	
5	|		0 + 5F/V	since V=4	
...	|		...		
V	-------		512	512	32
V + 1	-----		512 + F/V	512+32	32
V + 2	----		512 + 2F/V	512+64	32
V + 3	----		512 + 3F/V	512+96	416
V + 4			512 + 4F/V		
...			...		
2V			1024	1024	32
2V + 1			1024 + F/V	1024+32	32
2V + 2			1024 + 2F/V	1024+64	32
2V + 3			1024 + 3F/V	1024+96	416
...					
3V			1536	1536	32
3V + 1			1536 + F/V	1536+32	32
3V + 2			1536 + 2F/V	1536+64	32
3V + 3			1536 + 3F/V	1536+96	416

Within an FFT, the NCO will run continuously. Two mechanisms to start the NCO at the proper fringe phase are required. At bit F/V of each FFT cycle within the minor loop, the NCO state must be stored and recycled at the top of the next FFT cycle. At the end of the major loop, the store ram location will be later in the FFT cycle at point 512 - (F/V)*(V-1).

To see how this is accomplished on the FFT card, refer to k034d02.sch in Volume II. The FFT CARD RAM outputs the bits ST\0 and ST\1. ST\0 corresponds to the small numbers in the STORE DATA PT column. ST\1 corresponds to the large, less frequent number. In normal tracking, ST\0 and ST\1 are identical. The mux chooses which ST value to use. The stores are pipelined using the asic clock to deglitch the mux.

To see how the mux address is generated, refer to l002d32.sch in Volume II. This counter counts once per FFT cycle. An 8 bit radix from the control word sets the initial value. ADDRA` from the RCO is the LSB of the mux address. It chooses between ST\0 and ST\1. Chose the control word radix to equal hex 100 - V where V is the overlap factor given above. In the example on L002D32, V = 4, so 100 - 4 = FC. Thus when the counter counts V counts, the RCO is set to choose ST\1. Note this also reloads the radix.

3.3.2.4 Oversampled Data

When data is oversampled, the deformatter logic will alternate between driving the FFT engine with 512 even and 512 odd bits. If no overlapping is being performed, the NCO must run normally during the even bit FFT cycle, rewind to the even cycle fringe start phase for the start of the odd cycle, and then start the next even FFT cycle with the usual store rewind for the pause skip.

```
  S=STORE  R=REWIND
S R   EVEN SAMPLES   R    ODD SAMPLES   S R
```

If the system is also performing overlapping, the even and odd cycles will still be performed on adjacent FFT cycles. Thus there is a triple cycle operation. The third cycle is the even-odd cycle within the minor cycle. The minor cycle was initially represented in 3.3.2.3. Recall the minor cycle only occurs for fft sizes less than 512.

Oversampling is limited to a factor of 2, as far as the FFT card is concerned. Higher oversample factors are accomplished by throwing away some of the data in the PBI. Hence, the FFT card never sees the data. Higher oversample values would affect the overlap factor sent to the FFT card.

(OVERSAMPLED, 0A = EVEN, 0B = ODD)

EXAMPLE:
F=128, V=4, F/V=32

FFT CYCLE	FIRST DATA POINT TO FRINGE ROTATOR		STORE DATA PT
0A ---------->	0	0	NONE
0B ---------->	0	0	32
1A ---------->	0 + F/V	0+32	NONE
1B ---------->	0 + F/V	0+32	32
2A ---------->	0 + 2F/V	0+64	NONE
2B ---------->	0 + 2F/V	0+64	32
3A ---------->	0 + 3F/V	0+96	NONE
3A ---------->	0 + 3F/V	0+96	416
... |	...		
V A ----------	512	512	NONE
V B ----------	512	512	32
V + 1 A ---------	512 + F/V	512+32	NONE
V + 1 B ---------	512 + F/V	512+32	32

To see how the oversampling is accomplished on the FFT CARD, refer to k034d02.sch in Volume II. When the B address line of the mux is a 1, the ST\ line to the NCO's is a 1. Hence there will be no stores that FFT cycle.

Refer to L002d32.sch in Volume II, for the B address line generation. The LSB of the counter provides one signal. It is gated with the signal OVSAMP\ from the control word to generate the Address msb, ADDRB`. This is used to enable the oversampled mode. When there is oversampling, the Radix from the control word should equal: hex 100 - 2*V where V is the overlap factor.

4.0 FFT Bin Wiring

4.1 Introduction

Refer to the FFT Bin Schematics in Volume II. 1025d09.sch shows a top level view of the bin to bin wiring of the whole correlator.

Each FCC supplies trig tables to ten FFT cards. There is one FCC per rack.

4.2 Back Plane Buffer Boards

Due to noise problems, several versions of buffer boards were installed in the racks. The present rack configuration has evolved from successive attempts to deal with the noisy signals.

The problems arose from the FCC attempting to drive 5 FFT cards in parallel, combined with the FFT card not capturing the signal near the card edge.

The Backplane Buffer Boards are shown in 1025d40.sch in Volume II, as are the other 1025dxx.sch drawings referenced here. There are two boards per rack. The boards are bolted to the rear of the bin. The function of these boards is to take FCC signals that formerly drove five FFT cards, and provide five separate signals to the FFT cards. The signals buffered are CRSI (Control Register Serial In), CRSTB (Control Register Strobe), RAMCLK, and FRSI (Fringe Rotator Serial In).

4.3 FSTC Signal Distribution Problems

There were also problems with shifting in the Fractional Sample Time Correction (FSTC) data into the NCOs on the FFT Cards.

4.3.1 FSTC Original Signal Routing

The original routing of the Fractional Sample Data Serial In (FSSI) can be seen on the bottom of 1025d03.sch in Volume II. The FCC provided two signals that each went to five FFT cards, distributed to every other FFT card. Putting terminating resistors at each end of the chain was found to help. These resistors are also shown on 1025d03.sch. The resistors slip on the backplane pins.

The original routing of the Fractional Sample Shift Register Clock (FSSRCLK) can be seen in the lower right corner of 1025d01.sch in Volume II. The FCC provides five clocks, which serve adjoining pairs of FFT boards.

4.3.2 FSTC Buffer Board

In an attempt to fix the problems, the FSTC Buffer Board shown on 1025d42.sch in Volume II was developed. This board develops independent FSSI and FSSRCLK signals. That is each signal only drives one load. Note there are four FSSI signals per FFT board, one for each pipeline. These four FSSIs are clocked by a single FSSRCLK.

The FSTC Buffer Boards were installed in Racks 0 and 1. See 1025d45.sch in Volume II. The boards were placed on an unused DIN in the center of slot 6, next to the FCC. The installations required rewiring of the bin backplanes. On 1025d45.sch, the FSTC Buffer Board connectors are shown to the left of the FCC connectors.

However, there were still FSTC problems in the racks with the new FSTC Buffer Boards.

4.3.3 FSSRCLK FFT Card Modification

Eventually, the problem was found to be in the FSSRCLK distribution on the FFT card. The following modifications were made to each FFT card.

1. The card input pin FSSRCLK at P2C10, has its trace cut near the pin.

2. The original destination of FSSRCLK is grounded by a jumper placed, on the back of the FFT board, from U139 pin 1 to U139 pin 2. See top of 1002d02.sch in Volume II.

3. An HC541 was piggybacked onto U20 to buffer the FSSRCLK. Power and ground come from U20. Pin 1 of the piggyback gets a ground from pin 1 of U20. A jumper from pin 1 to 19 grounds pin 19 of the piggyback.

4. A wire is added from FSSRCLK at P2C10 to pins 2, 3, 4, and 5 of the piggyback.

5. The ASIC sockets were modified to isolate the FSSRCLK pin, on the stage 4 and 5 ASICs.

6. Wires were daisy chained from the U20 piggyback, directly to slip on connectors on the ASIC pins. As follows:

	Piggyback HC541 Pin	ASIC Pins
FSSRCLKA	18	U91 L5, U113 L5
FSSRCLKB	17	U47 L5, U70 L5
FSSRCLKC	16	U99 L5, U119 L5
FSSRCLKD	15	U53 L5, U78 L5

This was implemented on all FFT cards and solved the problems.

4.3.4 FSTC Buffer Boards Left in Place

The FSTC Buffer Boards were left in place in racks 0 and 1, as shown in 1025d45.sch and 1025d46.sch in Volume II. They were left in, since they improve signal quality. Plus, it would have been a lot of work to again rewire the racks. Racks 2 and 3 retain the original wiring, as shown in 1025d01.sch and 1025d02.sch.

5.0 The FFT Test Fixture

5.1 FFT Test Fixture Description

5.1.1 Block Diagram Description

Refer to The HCB Protocol for the FFT Test Fixture, fftfix.hcb, in Appendix V.

Refer to 1015d38.sch, the FFT Test Fixture Block Diagram, in Volume II.

The test fixture consists of 4 internal cards labeled A, B, C , and D. The top of the test fixture holds an FCC, FFT card, and MAC card.

A sequencer on Card B generates control signals. The sequencer is in two levels. The first level counts the 516 pulses of an fft cycle. The second level counts the number of fft cycles in a fringe and integration cycle. These drive roms to provide control signals.

The microprocessor loads a known data pattern into eight, 64k x 4 rams. These generate four, unique 2 bit data steams for the FFT card. The front panel switch allows selection of these data streams, or external data. Each unique, data stream is demultiplexed to the 4 inputs for each fft pipeline. The 2 bit data goes to the input selected by the address, ADR[0..1] output from the fft card. ADR[0..1] is selected by the card control word, shifted in with the asics. The exclusive or of the two data bits goes identically, to both bits of the other three data streams. Note, ADR[0..1] should then select the ram data stream via the FFT card input multiplexer.

The readout address to the ram is defined as follows. A fringe cycle is 256 fft cycles, equals 256*512 bits, equals 128K. The ram contains 256K. An integration cycle is 32 fringe cycles, equals 32*128K, equals 4096K. Thus the

ram is not big enough to hold an integration cycle's worth of random data. To pseudo randomize the data, the starting address of each fringe cycle read out progresses 4k each fringe cycle. Fringe cycle 0 starts being read out at address 0. Fringe cycle 1 begins at 4K. Fringe cycle 31 begins at 31*4K equals 124K, and ends at 124K + 128K equals 252K.

The sequencer provides the FCC an init. The FCC provides the trig tables and most of the control to the FFT Card. The FCC is down loaded via the HCB.

Each of the two FFT card output busses go to four of the MAC card inputs in parallel. The MAC card is normally controlled by the MCC. Since there is no MCC in the test fixture, the microprocessor and sequencer control the MAC card.

The mac card external address normally comes from the sequencer. This increments through all the ram locations for writing and accumulating. The address that is to be read out is from a register written by the microprocessor. The MACWE, during the 4 cycle gap, causes the external address to mux to this register. The microprocessor provides the address of the asic to be read on the MAC card via CRD SEL\, ROW0, ROW1, COL0, and COL1. AUXOUT selects the 18 bit half of the 36 bit word to be read from the asic. Short term accumulator read (STAREAD) clocks the address and output data into a register. The microprocessor reads the register.

5.1.2 FFT Test Fixture Files

The top Level Schematic is /corrdwgs/ffttfix/sch/SCCS/s.1015d37.sch. 1015d37.sch and 1015d38.sch are in Volume II. The remaining 1015dxx.sch can be found in the /corrdwgs/ffttfix/sch directory.

/corrdwgs/ffttfix/pals contains the PAL files.
/corrdwgs/ffttfix/proms contains the PROM files.
/corrdwgs/ffttfix/up contains the microprocessor code and HCB protocol, fftfix.hcb.

5.1.3 EXPOF\ in the FFT Test Fixture

The exponent overflow bits are latched and drive the front panel lights. They are reset by the master reset or the front panel button.

5.1.4 Clock Distribution in the FFT Test Fixture

The clock source can be selected by a three position front panel switch. The upper position, marked INT, selects the internal, voltage controlled oscillator (VCO). The center position selects a 32 MHz crystal. The lower position marked EXT, selects an external clock.

1015d21.sch shows the clock distribution schematic. Plugable delays select the clocks going to the various cards. 1015d20.sch shows the card to card timing.

5.2 Substitute MAC Card

See 1015d25.sch for the top level schematic. This is a large Shalloway card with a din connector on it. It mounts in the P2 slot of the MAC card position of the test fixture. The card compares the data from the FFT card to a known pattern stored in ram or rom and shows differences on the LEDs. The four rows of LEDs represent the 4 pipelines, with pipeline 0 on top, and the lsb's on the right.

The mini switches control the functionality. Switch positions of down equal zero. The romsel switch = 0 selects rams over roms. The roms were never tested due to difficulty in generating a data pattern. The page select switches only affect rom mode.

Ram mode occurs when compare4, compare2 = 00. Then pushing the capture button causes a snapshot to be taken in the ram. Subsequent errors cause the appropriate LEDs to light.

The latch switch causes the LEDs to latch on errors. The reset button resets the latch. Otherwise the errors flicker real time.

cmp4, cmp2 = 10 selects 512 compare mode. This is meant for 512 point or less FFTs where all 4 pipelines have identical results. On a bit basis, all 4 pipelines are compared. If they all agree, no LEDs light. If any of the 4 disagree, The actual value of all 4 bits are displayed.

cmp4, cmp2 = 01 selects 1K compare mode. Pipeline 0 is compared to pipeline 1. Any discrepancies causes the appropriate LED in row 0 to light. Pipeline 2 is compared to pipeline 3, with errors going to row 1.

The mode switch acts as follows:

When mode = 0, 8 fft cycles of data, continuously repeating, load the data ram with data repeating every 8 fft cycles. This cancels out the stagger in the data ram addressing, which increments 8 fft cycles or 4k each fringe cycle. This runs at a near 100% duty cycle. Lower fft repeat cycles can be had by changing data patterns. The fringe rotator and fstc would be off.

When mode = 1, compare only the first 128 ffts of data, giving more data patterns. This is ram only, since the rom is paged in 16 fft sections. The fringe rotator and fstc would be on. This has been the most used mode of the card.

5.3 Substitute FFT Card

A Substitute FFT Card exists for testing the MAC card independent of a FCC and FFT Card. See the schematic on L015D22.SCH. It consists of a sequencer, roms for a data pattern, and ECL drivers. There is a random data generator on board that can be selected by pulling the 7C255s and throwing switch 5A to RAND\. It plugs into DIN connector J4. J4 is the equivalent of P1, or the right had connector on the FFT Card.

5.4 New Snapshots for the FFT Test Fixture

The following script files use the indicated snapshots:

Script File	Snapshot
t512.cmd	512r.snap
t1k.cmd	1024r.snap
t2k.cmd	2048r.snap
t128.cmd	128r.snap
t2k.polar.cmd	2048r.polar.snap

To take a new snapshot, first run the appropriate .cmd file. Then issue the RTS commands:

```
takesnap 0
sendsnap "filename"
```

If "filename" is the above .snap name, then it will be used by subsequent calls to the .cmd file.

The .snap files are stored under SCCS in the diagnostics area. To update a snapshot, edit the snapshot, copy the updated .snap files from the delivery area, and then delget the snapshot. When people do a make, they will get the new snapshot.

5.5 Testing Strategies

v47a is downloaded from <load.tstfix. <t512.cmd provides a test using 512 point ffts. <t128.cmd tests 128 point ffts. <t1k.cmd tests 1k ffts. <t2k.cmd tests 2K ffts. <t2k.polar.cmd tests 2K ffts with mac polarization turned on. These tests provide a test pattern from the fcc through the mac card. <ffttst.cmd provides a comprehensive test of the different modes, including fft card input mux exercising.

The MAC Cards on the diagonals, containing the self products in the correlator, do not need to contain all the ASICs. For testing a partially populated MAC card, <tmacpartial.cmd is like <t512.cmd, only it does not stop for MAC control word loading failure.

The RTS command, senderr, displays the mac error pattern from the above tests. rsterr resets the error count and turns out the test fail light on the test fixture. Pushing the LED RESET button, on the test fixture, also turns out the light, but does not reset the error count. senderr also tells a probable bad FFT pipeline, if the error pattern is indicative of a bad pipeline.

To isolate faults on the fft card, run the t512.cmd, t1k.cmd, and t2k.cmd, noting the bad pipelines in each case. There is no criss-crossing of data paths for t512.cmd. There is criss-crossing into stages 0 and 4 for t1k.cmd, and criss-crossing into stages 0, 4, and 5 for t2k.cmd. If all 3 tests say a single pipeline, then stage 5 is probably at fault, since the fault was not criss-crossed into other pipelines. If t512.cmd and t1k.cmd say a single pipeline, while 2k says pipelines 0 and 2 or 1 and 3, then the fault is probably with stage 4 of the pipeline. If t2k.cmd shows all 4 pipelines bad, t1k.cmd shows 0 and 1, or 2 and 3 bad, and t512.cmd shows a single pipeline bad, which was also shown by t1k.cmd, then the fault is in stage 0, 1, 2, or 3 of the pipeline called out by t512.cmd. Suspect asics can be transferred to the mac card. The mac card display shows the specific asic that is bad.

If a stage 5 fault is not an asic, it may be the twiddle factor that is bad. If t512.cmd and t2k.cmd show the same single pipeline, but t1k.cmd does not show any errors, then the fault is probably with the large fstc rom logic, since that is not used in 1k mode. In 512 mode, stage 5 twiddles can be compared by adding a true and invert output, from the F399s, using the oscilloscope.

If t2k.cmd only has faults, the fault is probably with the angle mux logic in 2k mode (See 1002d06.sch and 1002d07.sch).

Sometimes certain data patterns will fail to create faults in one test that show up in another test. For example, t512.cmd might show no errors when another test fails.

The output of the fringe rotator roms is a good thing to compare with an oscilloscope, in t512.cmd mode. This tests the NCOs and the data input path.

The substitute MAC card can be used to see if it is a single bit of the output that is failing, in the case of stage 5 faults.

The test fixture will usually fail at around 38 MHz when the VCO clock is used.

The EXPOF\ lights also give an indication of which pipeline is failing.

The control words are checked as they shift out of the card to provide an indication, via the RTS, of control word load failure. Note in the correlator and testbed, the control words are not checked as they are shifted out.

If all tests show all pipelines bad with large numbers of errors, the fault could be with the clock, ram loading, trig tables, or due to multiple faults.

APPENDIX I HCB Protocol For The FFT Control Card

VLBA CORRELATOR PROJECT
NATIONAL RADIO ASTRONOMY OBSERVATORY
CHARLOTTESVILLE, VA 22903

Description (after 11/92)	Revision History	Date	By
	Preliminary	11/06/91	RPE
	Rev A	2/24/92	RPE
	Rev B	4/05/92	RPE
clarified Func 1D	Rev C	12/18/92	CMB
added Func 29 & 2A	Rev D	01/28/93	CMB
corrected adrs for Func 05	Rev E	05/06/93	CMB

Further revision history is stored via SCCS.

XFER TYPE	PROTOCOL	FREQ
NOP	00 ZZ (ZZ = DUMMY BYTE)	-
WRITE MEMORY	01 00 00 AA AA CC CC DD----DD	INIT (1)
READ MEMORY	02 00 00 AA AA CC CC	TEST
CALCULATE CHECKSUM	03 00 00 AA AA CC CC [2 BYTES RETURNED]	INIT (1)
TIME INIT	04 ZZ	TEST
RUN POLYNOMIAL TEST	05 ZZ	TEST
TIME INCREMENT	06 ZZ (ZZ = DUMMY BYTE)	TEST
TRIG TABLE	10 RR TT DD----DD	INIT (1)
TRIG TABLE CHECKSUM	11 RR [2 BYTES RETURNED]	INIT (1)
XILINX 1ST XFER	12 DD----DD (1389 BYTE XFER)	INIT (1)
XILINX 2ND XFER	13 DD----DD (1389 BYTE XFER)	INIT (1)
XILINX GO/NO GO FLAG	14 [1 BYTE RETURNED]	INIT (1)
ASIC CONTROL WORDS	15 AA DD----DD (96 BYTES OF DATA)	OBS (2)
ASIC CHECK SUM	16 AA [2 BYTES RETURNED]	OBS (2)
ASIC CONTROL WORD STROBE	17 ZZ (ZZ = DUMMY BYTE)	OBS (2)
FFT CARD RAM	18 AA DD----DD (1290 BYTES DATA)	OBS (2)
FFT CARD RAM	19 DD----DD (1290 BYTES DATA)	OBS (2)
FFT CARD RAM CHECK SUM	1A AA [2 BYTES RETURNED]	OBS (2)
PULSAR GATE RAM	1B TT DD----DD (1024 BYTES DATA)	OBS (2)
PULSAR GATE RAM CHECK SUM	1C ZZ (ZZ = DUMMY BYTE)	OBS (2)
FSTC MODES	1D 0D----0D (5 NIBBLES OF DATA)	OBS (2)
FSTC MODE CHECK SUM	1E [2 BYTES RETURNED]	OBS (2)
PBD ASSIGNMENT	1F SS PP	OBS (2)
PULSAR MODEL	20 MM PP--PP	AS NEEDED (3)
SPECIFY PULSAR MODEL	21 0M	AS NEEDED (3)
CRT PROMPT	22 AA----AA 00----00 (16 BYTES)	INIT (1)
START OBSERVATION	23 ZZ (ZZ = DUMMY BYTE)	OBS (2)
END OBSERVATION	24 ZZ (ZZ = DUMMY BYTE)	OBS
ASIC CONTROL WORD TEST	25 AA DD	TEST
TEST FRINGE/FSTC PARAMETERS	26 XY	SELF TEST
ZERO MODEL TIMES	27 ZZ (ZZ = DUMMY BYTE)	AS NEEDED (3)
READ MODEL TIME COUNTER	28 [8 BYTES RETURNED]	TEST
SELF TEST DETERMINED	29 ZZ (ZZ = DUMMY BYTE)	TO BE
READ MODEL	2A AA 0M (544 BYTES RETURNED)	TEST
MULTIPLE TIME INCREMENTS	2B AA	TEST
TRANSFER TIME	2C AA OM	TEST
Not currently used	2D	
READ PARAMETER MEMORY TIME	2E AA OM [8 BYTES RETURNED]	TEST
INCREMENT TIMES	2F ZZ (DUMMY)	TEST
READ PULSAR CHECKSUM	30 [2 BYTES RETURNED]	OBS

PROGRAM MODEL	80 AA 0M FF--DD (544 BYTES TOTAL)	AS NEEDED (3)
CHANGE ACTIVE MODEL	81 BB 0M	AS NEEDED (3)
CHANGE ALL MODELS	83 0M 0M 0M 0M 0M 0M 0M 0M 0M 0M	AS NEEDED (3)
PROGRAM TIMES	85 BB OM GG--GG	AS NEEDED (3)

The following table shows the name for the function code in the include file hcbFuncCodes.h. Also shown are the times in milliseconds from a survey of how long the functions take. Approximate data rates in KBytes/second are shown. Values are shown for when the fcc is and is not observing. There are potential problems for functions that have an equivalent transfer rate of less than 50 KBytes/sec, and take more than 10 ms. For a detailed explanation of the measurements
taken see /home/azalea1/staff/jgreenbe/notes/survey.txt. Functions labled INIT below are only used during initialization, where timing is not critical. The timing of functions labeled TEST is not of concern when observing. Data rates are not give where minimal data is transferred. Times in () are with the faster Dallas Microprocessor.

		OBSERVING		NOT OBSERVING	
INCLUDE FILE NAME	FC	TIME(ms)	RATE	TIME	RATE(KBytes/sec)
WRITETARGETMEM	0x01	INIT			
READTARGETMEM	0x02	TEST			
READCHKSUM	0x03	INIT			
CLRTIME	0x04	TEST			
EXECPOLY	0x05	TEST			
INCTIME	0x06	TEST			
LOADTRIG	0x10	INIT			
TRIGCHKSUM	0x11	INIT			
LD3030	0x12	INIT			
CONTINUELD3030	0x13	INIT			
LD3030GONOGO	0x14	INIT			
FFTCTRLWORD	0x15	46(10.2)	2.1	13.3(6.2)	7.2
FFTCTRLWRDCHK	0x16	Read from table so fast.			
STROBECTRLWORD	0x17	0.516		0.148	
LOADRAMS	0x18	Combined time for 18 and 19 below			
CONTINUELDRAMS	0x19	340	7.6	100	25.8
RAMCHKSUM	0x1a	Read from table so fast.			
LDPULSAR	0x1b	13.7ms(8.7)	74.9	13.7(8.7)	74.9
PULSARCHKSUM	0x1c	7000		2000 Note this returns fast, but has the microprocessor busy for the above times. 0x 30 below reads the results.	
FSTCMODES	0x1d	1.38(.541)	3.62	0.404(.351)	12.38
FSTCCHKSUM	0x1e	Read from table so fast.			
PBDTABLE	0x1f	0.54(.22)	3.70	0.15(.14)	13.3
PULSARMDLDWNLD	0x20	2.90(1.12)	30.3	1.09(.77)	80.7
PULSARMDLSELCT	0x21	0.52(.21)	1.9	0.15(.13)	6.7
FCCPROMPT	0x22	INIT			
STARTOBS	0x23	45.0(27.7)			
ENDOBS	0x24	0.15(0.13)			
SHIFTBYTE	0x25	TEST			
LDFRNGRATE	0x26	1.6(0.6)		1.6(0.6)	
ZEROMODELTIME	0x27	0.245(0.163)			
READMODELTIME	0x28	TEST			
SELFTEST	0x29	0.245(0.163)		0.148(0.133)	
MODELREAD	0x2a	TEST			
READPULSARCS	0x30	0.581(.265)	3.44	0.179(.166)	11.17
MODELPROGRAM	0x80	41.0(8.20)	13.27	6.53(3.3)	83.3
CHNGACTIVEMDL	0x81	0.516(0.226)		0.150(0.138)	
CHNGALLMODELS	0x83	0.755(.292)	13.25	0.219(.184)	N45.66
PROGPMTIME	0x85	0.821(.318)	11.0	0.239(.201)	37.66

(1) mode independent command issued upon power up or for system initialization.
(2) mode dependent command (3) operational command

5) ASIC CONTROL WORDS

```
ASIC CONTROL WORDS:          15 AA DD----DD (96 BYTES OF DATA)
ASIC CHECK SUM:              16 AA [2 BYTES RETURNED]
ASIC CONTROL WORD STROBE:    17 ZZ (ZZ = DUMMY BYTE)
```

where

AA is a card assignment (00 to 09 in the system, 10 in the FFT test fixture).

DD----DD Indicates 96 bytes of data (four bytes per ASIC. Bytes within ASIC are sent LS byte first. ASIC are sent in the reverse order that they occur in the ASIC control word serial string, i.e., chip U40 first, U43 next, ... and U106 last).

Checksum is made as bytes are received from the HCB. The checksum for each card (AA) are stored in a table, so there is no restriction on when the checksum must be requested.

6) FFT CARD RAM TABLES

```
FFT CARD RAM TABLES:          18 AA DD----DD (1290 BYTES DATA)
FFT CARD RAM TABLES:          19    DD----DD (1290 BYTES DATA)
FFT CARD RAM TABLE CHECK SUM: 1A AA [2 BYTES RETURNED]
```

where

AA is a card assignment (00 TO 09)

DD----DD indicates 1290 bytes of data (see figure 1)

Checksum is made as bytes are received from the HCB. The checksum for each card (AA) are stored in a table, so there is no restriction on when the checksum must be requested.

7) PULSAR GATE TABLE

```
PULSAR GATE TABLE:            1B TT DD----DD (1024 BYTES DATA)
PULSAR GATE TABLE CHECKSUM:   1C ZZ [ZZ=DUMMY BYTE]
READ PULSAR CHECKSUM          30 [2 BYTES RETURNED]
```

where

TT goes from 00 to 7F for the 128 transactions required to fill the pulsar gate RAM.

DD----DD indicates 1024 bytes of data per transaction. The pulsar gate RAM is 4-bits wide and each DD byte carries two nibbles. Total RAM locations filled is 2048 X 128 or 262144.

128 separate calls to hcb1b download the pulsar gate ram. After the ram has been downloaded, hcb1c can be called to generate a checksum for the entire ram. The checksum is stored in memory by hcb1c, and can be retrieved by hcb30 as two bytes. A separate retrieval step is necessary, since it takes so long to calculate the checksum, that the hcb bus would hang while waiting for the result. Enough time must pass after hcb1c, to allow the checksum calculation to finish. Otherwise, issuing another hcb command might cause the the hcb bus to hang. This time is about 7 seconds if we are observing, or 2 seconds if we are not observing (as determined by startobs and endobs commands).

Both loading the ram and calculating the checksum, put the rams in program mode, as determined by the common PROG\ signal. Program mode temporarily disrupts the trig table output. Thus a job should not be running while using either of these functions. Since the trig tables are trashed anyway, we might as well endobs before doing these operations, so that they run faster.

---

1) NOP, MAIN MEMORY

NOP:	00 ZZ (ZZ = DUMMY BYTE)
WRITE MEMORY:	01 00 00 AA AA CC CC DD----DD
READ MEMORY:	02 00 00 AA AA CC CC
CALCULATE CHECKSUM:	03 00 00 AA AA CC CC [2 BYTES RETURNED]

where

00 00 AA AA is a 32-bit start address and CC CC is a transfer count with msbyte first. Checksum returned is a 16-bit sum of all bytes in the command range.

2) MISCL TEST

TIME INIT:	04 ZZ
RUN POLYNOMIAL TEST:	05 ZZ
TIME INCREMENT:	06 ZZ (ZZ = DUMMY BYTE)

The TIME INIT command resets the model time counter (immediately as opposed to at a set time as the function code 27 command does).

For the RUN POLYNOMIAL TEST command, 6 IEEE double precision parameters are taken from memory location 1FC0. The IEEE dp result is left at memory location 1FF0.

3) TRIG TABLE

TRIG TABLE:	10 RR TT DD----DD
TRIG TABLE CHECKSUM:	11 RR [2 BYTES RETURNED]

where:

STAGE	RR	TT	XFER COUNT
1	90	00 AND 01	1024
1	91	00 AND 01	1024
2	92	00 THRU 07	1024
2	93	00 THRU 07	1024
3	94	00 THRU 31	1024
3	95	00 THRU 31	1024
3	96	00 THRU 31	1024
4	98	00 THRU 63	1024
4	99	00 THRU 63	1024
4	9A	00 THRU 63	1024
5	9B	00 THRU 63	1024
5	9C	00 THRU 63	1024
5	9D	00 THRU 63	1024

Check sum is a 16-bit result made from information stored in the trig table RAMs.

4) XILINX PERSONALITY

FIRST XFER:	12 DD----DD (1389 BYTE XFER)
SECOND XFER:	13 DD----DD (1389 BYTE XFER)
GO/NO GO FLAG:	14 (1 BYTE RETURNED)

A byte = FF is returned if the Xilinx load was successful, 00 if unsuccessful.

8) FSTC MODES (programs the FSTC pals 7E and 7F for FFT size via RAM 1E. Though these signals enter the FSTC Xilinx, they do not affect its functionality)

```
FSTC MODES:                    1D 0V 0W 0X 0Y 0Z
FSTC MODE CHECK SUM:           1E [2 BYTES RETURNED]
```

where

```
0V = 0000 bbaa      bb = FFT card 1 code, aa = FFT card 0 code
0W = 0000 ddcc      dd = FFT card 3 code, cc = FFT card 2 code
0X = 0000 ffee      ff = FFT card 5 code, ee = FFT card 4 code
0Y = 0000 hhgg      hh = FFT card 7 code, gg = FFT card 6 code
0Z = 0000 jjii      jj = FFT card 9 code, ii = FFT card 8 code
```

the 2-bit codes aa through jj are of the form;

xx = 00 for short FFT
xx = 01 for 1024 point FFT
xx = 10 for 2048 point FFT

Check sum is made as bytes are received from the HCB. The checksum is stored in ram, so there is no restriction on when the checksum must be requested.

An alternative representation of the above information is presented below:

7	6	5	4	3	2	1	0	
0	0	0	0	B1	B0	B1	B0	
				card #		card #		
				1		0		byte 0
				3		2		byte 1
				5		4		byte 2
				7		6		byte 3
				9		8		byte 4

For each FFT Card, the two bit field B1,B0 is defined as follows:

B1	B0	FFT SIZE
0	0	Less than 1K
0	1	1K
1	0	2K

9) PBD TABLE ASSIGNMENT

```
PBD ASSIGNMENT:                1F SS PP
```

where

SS is the station assignment (0 TO 9)
PP is the PBD assigned to FFT card SS (0 TO 13)

This has the fcc send the correct integer delay to the appropriate deformatter, per the star diagram. This must correspond to the mux positions sent to the fft cards, via the asic control words. Also the mcc reproduces the star diagram, for the validities.

10) PULSAR MODEL COEFFICIENTS

```
PULSAR MODEL:                  20 0M PP--PP
CHANGE ACTIVE PULSAR MODEL:    21 0M
```

0M is a model assignment (0 to 3)
PP--PP is the pulsar model parameters (88 bytes per model)

11) CRT PROMPT MESSAGE

```
CRT PROMPT:                    22 AA---AA 00---00 (16 BYTES)
```

Up to 15 ASCII bytes AA---AA are transferred. The remaining byte(s) 00---00 are needed to get the total data portion of the command to 16 (the total transfer is 17-bytes, the one byte function code plus 16 data bytes). The ASCII portion of the transfer is added to the CRT prompt after "FFT CONTROL CARD".

12) OBSERVATION CONTROL

```
START OBSERVATION:             23 ZZ
END OBSERVATION:               24 ZZ (ZZ = DUMMY BYTE)
```

13) ASIC CONTROL WORD TEST

```
ASIC CONTROL WORD TEST:        25 AA DD
```

where

AA is a card assignment (00 to 09 in the system, 10 in the FFT test fixture).
DD indicates 1 byte of data

14) LOAD TEST FRINGE AND FSTC PARAMETERS

```
LOAD TEST FRINGE/FSTC PARAMETERS:    26 XY
```

where

for X = 0, FFT size is 64 to 512 point
for X = 1, FFT size is 1024 point
for X = 2, FFT size is 2048 point
for Y = 0, INITIAL FRINGE PHASE = ZERO
FRINGE RATE = ZERO
INITIAL FSTC SLOPE = ZERO
FSTC RATE = ZERO

for Y = 1, INITIAL FRINGE PHASE = 1122334455 (40-BITS)
FRINGE RATE = 123456789A (40-BITS)
INITIAL FSTC SLOPE = 54321 (20-BITS)
FSTC RATE = 12345 (20-BITS)

Note: The fringe parameters are stored in the FFT card NCO and are inaccessible. The FSTC parameters, however, are stored in memory as below and are hence accessible for modification after issuance of the load fringe command;

FFT PIPELINE 0 AND 1	FFT PIPELINE 2 AND 3 *	DATA FOR X = 1
7000	7008	54
7001	7009	32
7002	700A	10
7003	700B	00
7004	700C	12
7005	700D	34
7006	700E	50
7007	700F	00

* for 2048 point transforms only 7000 to 7007 used

15) Zero Model Time

27 ZZ (ZZ = DUMMY BYTE)

This zeros the ten model zero times in parameter memory. It does this by setting a flag. In the TIME function while observing, if the flag is detected, on a tic boundary the times are reset.

16) READ MODEL TIME COUNTER

READ MODEL TIME COUNTER: 28 [8 BYTES RETURNED]

Will read the 4-msec time from microprocessor memory. The value returned is a double precision IEEE number and is the value of the time tick counter used for station 9 at the start of the last 131 msec integration cycle tick before reception of the command. This function is of limited usefulness since the time storage was moved to parameter memory. If it is not used at all, support for it can possibly be eliminated from the observe loop. That is the function RDTIME could be eliminated in obs.asm.

17) SELF TEST

SELF TEST: 29 ZZ (ZZ = DUMMY BYTE)

Instructs the FCC to run the 4 msec self test at the beginning of the next 131 msec interval. Details of test to be defined later.

18) READ MODEL TEST

READ MODEL: 2A AA OM (544 BYTES RETURNED)

where

AA is the FFT card number (0-9)
0M is the model bank (0-3)

and the returned bytes are:

F0--F0 F1--F1 F2--F2 F3--F3 D0--D0 D1--D1

where

F0--F0 are channel 0 fringe parameters (88 bytes per fringe model)
F1--F1 are channel 1 fringe parameters
F2--F2 are channel 2 fringe parameters
F3--F3 are channel 3 fringe parameters
D0--D0 are phase center 0 delay parameters (96 bytes per model)
D1--D1 are phase center 1 delay parameters

19) MULTIPLE TIME INCREMENTS 2B AA TEST

Advance the 4ms tic counter by AA * 65536 increments. The hcb will hang on subsequent commands until this is done.
hcbOffsetFccTime(rack,offset) can be used for offsets of less than 64k.

20) TRANSFER TIME 2C AA OM TEST
AA is the station number (0 TO 10, 10=pulsar)
0M is a model assignment (banks 0 TO 3)
Copy the time stored in parameter memory location AA OM into the floating point chip time register (rf0). Note this can be used in conjunction with PROGRAM TIME, hcb85, to write a desired value into the fpc.

21) Not currently used 2D

22) READ PARAMETER MEMORY TIME 2E AA OM [8 BYTES RETURNED] TEST
AA is the station number (0 TO 10, 10=pulsar)
0M is a model assignment (banks 0 TO 3)
The value returned is a double precision IEEE number and is the value of the time tick counter at that location in parameter memory. Note this might have occasional errors returned if done while observing.

23) INCREMENT TIMES 2F ZZ (DUMMY) TEST
Increment the time for active model for each of the ten stations. The times will be incremented in parameter memory. Normally times are incremented as part of the observing loop.

24) READ PULSAR CHECKSUM 30 [2 BYTES RETURNED] OBS
Read the pulsar gate ram checksum calculated by a previous call to hcb1c.

25) FRINGE AND DELAY MODEL COEFFICIENTS
These are fast hcb functions, as indicated by the 80 bit.

PROGRAM MODEL: 80 AA 0M FF--FF FF--FF FF--FF FF--FF DD--DD DD--DD
CHANGE ACTIVE MODEL: 81 BB 0M
CHANGE ALL MODELS: 83 0M 0M 0M 0M 0M 0M 0M 0M 0M 0M
PROGRAM TIMES 85 BB OM GG--GG
Copy the GG--GG time into the parameter memory location specified by BB OM

89, 91, A1, C1 would be subsequent addresses, based on the current scheme.

where
AA is the station number (0 TO 10) 10= selftest model, only bank 0 available
BB is the station number (0 TO 10) 10=pulsar
0M is a model assignment for changing the active model (banks 0 TO 3)
FF--FF are channel 0 fringe parameters (88 bytes per fringe model)
FF--FF are channel 1 fringe parameters
FF--FF are channel 2 fringe parameters
FF--FF are channel 3 fringe parameters
DD--DD are even phase center delay parameters (96 bytes per delay model)
DD--DD are odd phase center delay parameters (544 bytes total)
0M--0M are 10 model assignments for the 10 stations (1st 0M after function code is for station 0, last 0M byte received is for station 9).
GG--GG is the time (8 bytes) to be written in parameter memory location 160H + OM*20H + AA*2

Note in hcb80, the model is initially stored in microprocessor memory, then downloaded to the parameter memory. If observing, hcb80 will take 8 additional 4ms tics to complete the downloading. This extra time should be allowed for. It will be especially noticeable if you are trying to download multiple stations in succession.

Model assignments become effective on the 131 msec integration cycle tick following reception of the HCB command.

The fringe/pulsar model parameter format is seen below;

ADDRESS IN PARAMETER MEMORY	PARAMETER	FUNCTION
0000	P5	5th order phase polynomial coefficient
0002	P4	4th order phase polynomial coefficient
0004	P3	3rd order phase polynomial coefficient
0006	P2	2nd order phase polynomial coefficient
0008	P1	1st order phase polynomial coefficient
000A	P0	0th order phase polynomial coefficient
000C	R4	4th order rate polynomial coefficient
000E	R3	3rd order rate polynomial coefficient
0010	R2	2nd order rate polynomial coefficient
0012	R1	1st order rate polynomial coefficient
0014	R0	0th order rate polynomial coefficient
0016		not used

The delay model parameter format is seen below:

ADDRESS IN PARAMETER MEMORY	PARAMETER	FUNCTION
0000	P5	5th order phase polynomial coefficient
0002	P4	4th order phase polynomial coefficient
0004	P3	3rd order phase polynomial coefficient
0006	P2	2nd order phase polynomial coefficient
0008	P1	1st order phase polynomial coefficient
000A	P0	0th order phase polynomial coefficient
000C	M0	mode dependent shift parameter
000E	R4	4th order rate polynomial coefficient
0010	R3	3rd order rate polynomial coefficient
0012	R2	2nd order rate polynomial coefficient
0014	R1	1st order rate polynomial coefficient
0016	R0	0th order rate polynomial coefficient

where M0 delay model parameter is defined as:

TRANSFORM	M0 (DOUBLE PRECISION INTEGER)
64 POINT	0000 0000 0000 0000 (0)
128 POINT	0000 0000 0000 0001 (1)
256 POINT	0000 0000 0000 0002 (2)
512 POINT	0000 0000 0000 0003 (3)
1024 POINT	0000 0000 0000 0004 (4)
2048 POINT	0000 0000 0000 0005 (5)

ADDENDUM I of APPENDIX I - FCC Models and Timing

In order to support model tracking in one station, an FFT control card requires 4 sets of fringe model parameters (one for each of 4 channels), 2 sets of delay model parameters (one for each of 2 phase centers), and 1 set of pulsar model parameters. Each model requires 11 double precision IEEE floating point coefficients (delay models require an additional integer).

There is storage room on an FFT control card for 4 complete sets of model parameters for all 10 stations controlled by an FFT control card. While the system is observing, each station is assigned an active model from among the 4 possible sets of model parameters in storage. This active model is the one used to track the models in the correlator at any given time.

The hardware model generator will use the active model parameters for each of the 10 stations to compute polynomials with a time variable, the results of which will program hardware stages in the VLBA correlator that perform the actual model related tasks. The hardware model generator evaluates a complete set of model polynomials (4 fringe, 2 delay and 1 pulsar) for each of the 10 stations every 4.096 msec cycle.

The time variable used in the computation of the model polynomials is a floating point number that increments by one for every 4.096 msec time tick. These are stored as double precision floating point numbers in parameter memory.

In parameter memory, there are 11 active time variables: one for each fft card or station, and one for the pulsar model. There are 4 sets of these. The one that is currently used corresponds to the active model specified for that station or pulsar model.

The multiple time variables allows separate subarrays to keep their own time. Having a single time for all caused loss of precision in evaluating models when times got large.

The time is sent to parameter memory by:

```
      PROGRAM TIMES       85 BB OM GG--GG
            Copy the GG--GG time into the parameter memory
            location specified by BB OM
       BB is the station number  (0 TO 10) 10=pulsar
       0M is a model assignment  (banks 0 TO 3)
This can be initialized to any desired value.
See        hcbWritePmTime(station,glmodelnr,stnewtime,rack,fcc)
```

Note that if observing, the times for the active model will be incrementing, as can be observed by:

```
22)     READ PARAMETER MEMORY TIME 2E AA OM [8 BYTES RETURNED]  TEST
        AA is the station number  (0 TO 10, 10=pulsar)
        0M is a model assignment  (banks 0 TO 3)
```

The value returned is a double precision IEEE number and is the value of the time tick counter at that location in parameter memory. Note this might have occasional errors returned if done to an active bank while observing.

The above is called by: `readPmTimes(model,busnr,targetnr)`

The correct procedure to update the times is to send the time to an inactive model bank, then change active models to that bank. Make sure that initially the times are not being sent to an active bank, or hey will each start incrementing as soon as they are put in, if you are observing.

Actual model parameters are transmitted to the FFT control card through the use of 2 HCB commands;

```
PULSAR MODEL:                20 0M PP--PP (88 PP BYTES PER MODEL)
PROGRAM MODEL:               80 AA 0M FF--FF FF--FF FF--FF FF--FF DD--DD DD--DD
```

These commands will transfer the IEEE parameters calculated by the RT computer system to the FFT control card. Parameters should not be sent to the active model of a station since the transmission time to correlator cards over the HCB has a 131 msec time uncertainty. The correct way to update station model parameters is to send the new coefficients into an inactive model. Model parameters may thus be updated as frequently as every 131 msec or as seldom as desired.

Parameters sent into an inactive model will remain unused in the FFT control card RAM storage until activated by one of the HCB commands below:

To change active models use:

To change a single station use:

```
      CHANGE ACTIVE MODEL:  81 BB 0M
BB is the station number  (0 TO 10) 10=pulsar
0M is a model assignment   (0 to 3)
```

To change all 10 stations use:

```
       CHANGE ALL MODELS:    83 0M 0M 0M 0M 0M 0M 0M 0M 0M 0M
```

See: chngallmodels(glmodelnr,rack,fcc)

There is one pulsar model for all ten stations. To change its active model use:

```
         CHANGE ACTIVE PULSAR MODEL:    21 0M
      Note 0x81 0xA 0M has the same effect as 0x21 0M
```

See: chngactivepulsarmdl(glmodelnr,rack,fcc)

These HCB commands will change the active model for one or all stations to the model indicated by the command. The actual change will occur on the first 4.096 msec interrupt of the 131 msec integration cycle following reception of the command. These commands may also be used to support model switching.

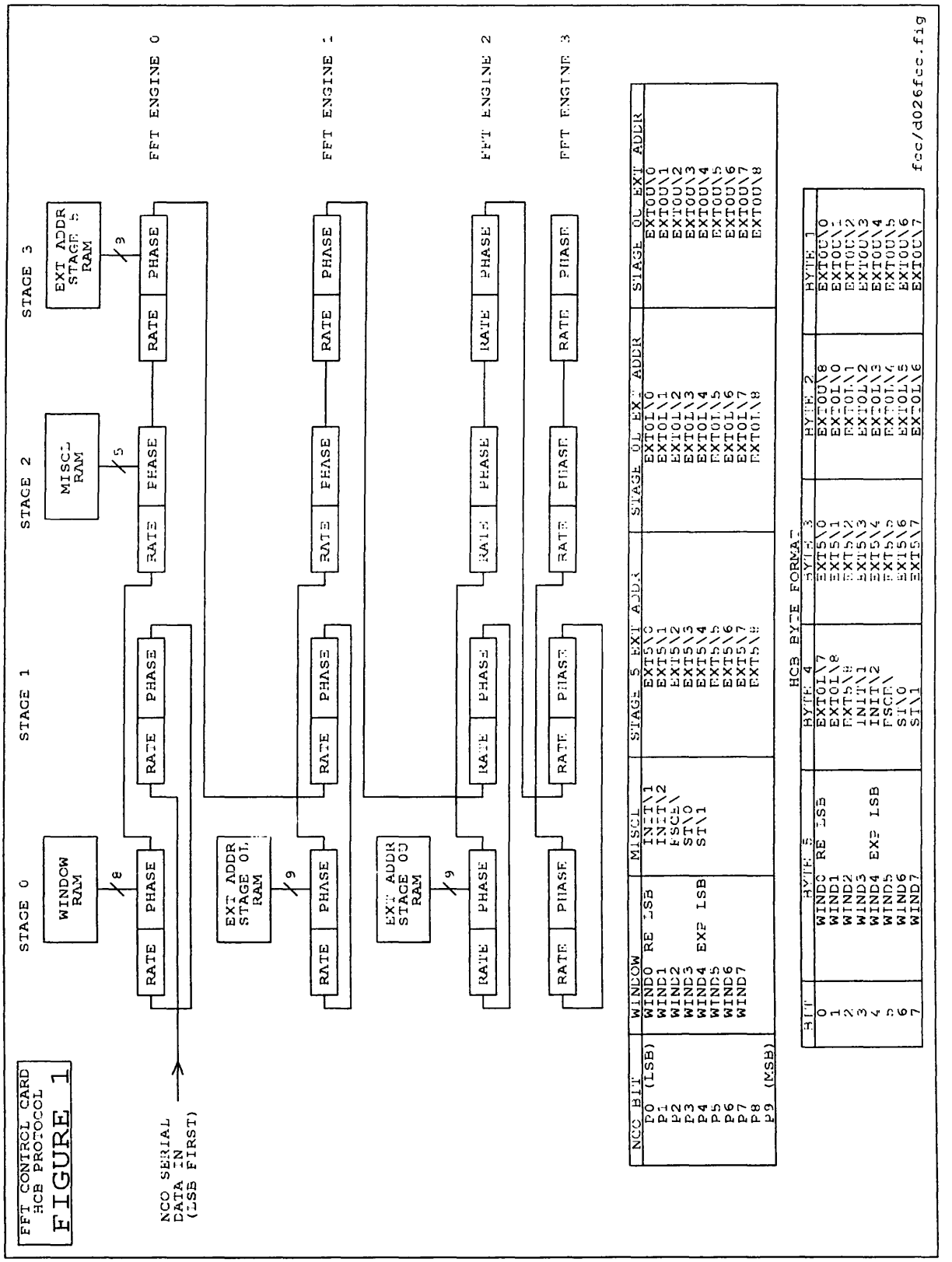


Figure 1 FFT Control Card HCB Protocol

APPENDIX II Text PAL Files

FSTC Serial Word Mux 7F & 7E Abel Text File
In Volume II, there are Orcad PAL Layout Schematic Representations. The Abel text files are included as part of the schematics. This section was too long to fit on its schematic, so it is included here.

FFT Card PALASM Files
These files have never been converted to Abel, since no additional work has been done on them.

pal7f.abl FSTC Serial Word Mux 7F & 7E TEXT FILE

The text file, PAL7F.ABL FILE, is included here, since there was not room on the included Orcad drawing, PAL7F.SCH (Sheet 5).

```
module PAL_7F

title 'FSTC DATA MUX PAL

Ray Escoffier   19 August, 1989'

                U7F      device          'P22V10' ;

"       This is a replacement of the  original 16v8 version         "

" PAL7F drives the FSTC serial data to stations 0, 2, 4, 6 and 8    "
" for pipelines 0-3 (or pipelines 4-7 depending on rack)            "
" See /corrdwgs/fcc/sch/k027d02.blk for signals from Xilinx "
" See fccasm/fcc.doc and fccasm/d030t01.mod "

"inputs......................................................"

                FSTC0           PIN 4  ;
                FSTC1           PIN 5  ;
                FSTC2           PIN 6  ;
                FSTC3           PIN 7  ;
                FSTC4           PIN 8  ;
                FSTC5           PIN 9  ;
                FSTC6           PIN 10 ;
                FSTC7           PIN 11 ;
                M0              PIN 21 ;
                M1              PIN 20 ;
                FFT0            PIN 15 ;
                FFT14           PIN 14 ;

"internal storage, not used externally............................."

              SIGN0         PIN 23 ;
              SIGN1         PIN 22 ;

"SIGN0 latches the sign bit to be used for sign extension for FSSIB "
"in <=512 and 1K and for FSSIB, FSSIC and FSSID in 2K               "

"SIGN1 latches the sign bit to be used for sign extension for FSSID "
"in <=512 and 1K                                                    "


"    M1     M0      MODE                                  "
" ------------------------------------------------        "
"     0      0     64, 128, 256, OR 512 POINT FFT         "
```

```
"    0     1    1024 POINT FFT                    "
"    1     0    2048 POINT FFT                    "
"                                                                "
"     FFT0  = 1 PRODUCES SIGN EXTENSION                          "
"     FFT14 = 0 DURING 10-BIT SHIFTOUT OF THE FSTC NCO PHASE     "
"     FFT14 = 1 DURING 10-BIT SHIFTOUT OF THE FSTC NCO RATE      "

"outputs......................................................"

              !FSSIA          PIN 19 ;  "pipeline 0"
              !FSSIB          PIN 18 ;  "pipeline 1"
              !FSSIC          PIN 17 ;  "pipeline 2"
              !FSSID          PIN 16 ;  "pipeline 3"

"definitions.................................................."

               MODE = [M1,M0]                                ;

"
"equations...................................................."

equations

      SIGN0    :=         0  &  (MODE == [0,0])  & !FFT14 & !FFT0   "PHASE"
                #  FSTC2  &  (MODE == [0,0])  &  FFT14 & !FFT0   "RATE"

                   #  FSTC0  &  (MODE == [0,1])  & !FFT14 & !FFT0   "PHASE"
                #  FSTC4  &  (MODE == [0,1])  &  FFT14 & !FFT0   "RATE"

                   #      0  &  (MODE == [1,0])  & !FFT14 & !FFT0   "PHASE"
                #  FSTC4  &  (MODE == [1,0])  &  FFT14 & !FFT0   "RATE"

                #  SIGN0  &                              FFT0   ;

      SIGN1    :=         0  &  (MODE == [0,0])  & !FFT14 & !FFT0   "PHASE"
                #  FSTC3  &  (MODE == [0,0])  &  FFT14 & !FFT0   "RATE"

                   #  FSTC1  &  (MODE == [0,1])  & !FFT14 & !FFT0   "PHASE"
                #  FSTC5  &  (MODE == [0,1])  &  FFT14 & !FFT0   "RATE"

                   #      0  &  (MODE == [1,0])  & !FFT14 & !FFT0   "PHASE"
                #  FSTC4  &  (MODE == [1,0])  &  FFT14 & !FFT0   "RATE"

                #  SIGN1  &                              FFT0   ;

"PIPE 0"
"for FSSIA, sign extension, using the sign of the rate, is only applied to"
"the rates for all fft sizes"
"since the phase is all zeros, we do not want the sign of the rate applied"
"to the phase for any of the three FFT sizes for FSSIA (pipe 0)          "


      FSSIA    :=        0   &  (MODE == [0,0])  & !FFT14  & !FFT0  "PHASE"
                   #  FSTC2  &  (MODE == [0,0])  &  FFT14  & !FFT0  "RATE "

                   #    0    &  (MODE == [0,1])  & !FFT14  & !FFT0  "PHASE"
                   #  FSTC4  &  (MODE == [0,1])  &  FFT14  & !FFT0  "RATE "

                   #    0    &  (MODE == [1,0])  & !FFT14  & !FFT0  "PHASE"
                   #  FSTC4  &  (MODE == [1,0])  &  FFT14  & !FFT0  "RATE "

                   #    0                        & .FFT14  &  FFT0  "PHASE"
                #  SIGN0                      &  FFT14  &  FFT0; "RATE "
```

```
"PIPE 1"
"sign extension, using the sign of the rate, is applied to both phase and "
"rate except for the phase for fft sizes <= 512 and 2K, which are zero    "

"The Xilinx FSTC7 term is faulty.  FSTC7 adds NCOx1 plus NCOx512 for"
"2K pipe 3 initial phase.  FSTC7 needs sign extension on the NCOx1"
"term.  In the 2K phase equation, we are changing FSTC7 to FSTC6 so"
"pipe 3 will have the same initial phase as pipe 2.  This is done as an"
"alternative to fixing the Xilinx.  The error of the initial phase"
"being slightly off is not considered significant.  For consistency the"
"initial phases are made identical for pipes 0 and 1 also."
"This entailed changing the 2K Phase from FSTC0 to 0. "

      FSSIB    :=        0   &  (MODE == [0,0])  & !FFT14  & !FFT0  "PHASE"
                   #  FSTC2   &  (MODE == [0,0])  &  FFT14  & !FFT0  "RATE "

                   #  FSTC0   &  (MODE == [0,1])  & !FFT14  & !FFT0  "PHASE"
                   #  FSTC4   &  (MODE == [0,1])  &  FFT14  & !FFT0  "RATE "

                   #     0    &  (MODE == [1,0])  & !FFT14  & !FFT0 "PHASE 2K"
                   #  FSTC4   &  (MODE == [1,0])  &  FFT14  & !FFT0 "RATE 2K "

                   #     0    &  (MODE == [0,0])  & !FFT14  &  FFT0  "PHASE 512"
                   #  SIGN0   &  (MODE == [0,0])  &  FFT14  &  FFT0  "RATE  512"
                #  SIGN0   &  (MODE == [0,1])  &            FFT0  "P & R 1K "
                   #     0    &  (MODE == [1,0])  & !FFT14  &  FFT0  "PHASE 2K "
                   #  SIGN0   &  (MODE == [1,0])  &  FFT14  &  FFT0; "RATE 2K "


"PIPE 2"
"for FSSIC, we want the rate sign bit used for sign extension for rates in"
"all three FFT cases; we want zeros for the upper five bits for 512 and 1K"
"and we want to pass fstc6 thru during the upper 5 bits for 2K"
"for 512 and 1K, sign1 is the storage element to use"
"for 2K, sign0 is same as sign1 and we still use sign1 for the rate"


      FSSIC    :=        0   &  (MODE == [0,0])  & !FFT14  & !FFT0  "PHASE"
                   #  FSTC3   &  (MODE == [0,0])  &  FFT14  & !FFT0  "RATE "

                   #    0     &  (MODE == [0,1])  & !FFT14  & !FFT0  "PHASE"
                   #  FSTC5   &  (MODE == [0,1])  &  FFT14  & !FFT0  "RATE "

                   #  FSTC6   &  (MODE == [1,0])  & !FFT14           "PHASE"
                   #  FSTC4   &  (MODE == [1,0])  &  FFT14  & !FFT0  "RATE "

                   #     0    &  (M1 == 0      )  & !FFT14  &  FFT0  "P 512&1K"
                #  SIGN1   &                       FFT14  &  FFT0; "ALL RATES"

"PIPE 3"
"for FSSID we want sign extension, using the sign of the rate, for all rates"
"we need the same sign extension for phase in 1K"
"we want zeros for <=512 phase"

"The Xilinx FSTC7 term is faulty.  FSTC7 adds NCOx1 plus NCOx512 for"
"2K pipe 3 initial phase.  FSTC7 needs sign extension on the NCOx1"
"term.  In the 2K phase equation, we are changing FSTC7 to FSTC6 so"
"pipe 3 will have the same initial phase as pipe 2.  This is done as an"
"alternative to fixing the Xilinx.  The error of the initial phase"
"being slightly off is not considered significant.  For consistency the"
"initial phases are made identical for pipes 0 and 1 also."
```

```
FSSID    :=        0   &  (MODE == [0,0])  & !FFT14  & !FFT0  "PHASE"
             #  FSTC3  &  (MODE == [0,0])  &  FFT14  & !FFT0  "RATE "

             #  FSTC1  &  (MODE == [0,1])  & !FFT14  & !FFT0  "PHASE"
             #  FSTC5  &  (MODE == [0,1])  &  FFT14  & !FFT0  "RATE "

             #  FSTC6  &  (MODE == [1,0])  & !FFT14     "2K PHASE"
             #  FSTC4  &  (MODE == [1,0])  &  FFT14  & !FFT0  "RATE "

             #     0   &  (MODE == [0,0])  & !FFT14  &  FFT0  "PHASE 512"
             #  SIGN1  &                      FFT14  &  FFT0  "ALL RATES"
             #  SIGN1  &  (MODE == [0,1])  & !FFT14  &  FFT0; "PHASE 1K"


"The following is the history behind the current (as of July 24, 1997)      "
"equations:                                                                 "
"                                                                           "
"We need to do sign extension for the 1K and 2K modes in the pals at        "
"7E/7F on the FCC.  In some cases, the sign extension is required in        "
"the phase as well as in the rate. NO sign extension for 2K, pipes 2 & 3    "
"phase, let the input bits pass thru without gating of any kind             "
"We need to determine how many storage flip-flops are required in each      "
"pal.  There are two free outputs for use as storage in 7E, and one in      "
"7F.  By removing Vcc from 7F22, there can be two free outputs in 7F        "
"also.                                                                      "
"                                                                           "
"In all cases where the PHASE is not a hard wired zero, the PHASE is        "
"a simple function of the RATE.  The only three functions are:              "
"                                                                           "
"      PHASE = RATE/2        1K, pipes 1 and 3                             "
"                                                                           "
"      PHASE = RATE*256      2K pipe 2                                     "
"or                                                                         "
"      PHASE = RATE*256.5    2K pipe 3 (This has been eliminated)          "
"                                                                           "
"In each case above, the sign of PHASE is always the same as the sign       "
"of RATE.  "
"                                                                           "
"In all cases where phase is hard wired zero, the sign extension must"
"not copy the rate sign to the phase sign, but must leave all phase"
"bits zero.  Such sign extension creates the erroneous value of"
"1111 1000 0000 0000 0000 for negative numbers."
```

```
"                                                                              "
"The following table is used to determine how many independent storage         "
"elements are required:  (the table shows for each output {FSSIx} which       "
"input {fstcx} is the source of the phase or rate parameter)                   "
"                                                                              "
"         FSSIA          FSSIB          FSSIC          FSSID                  "
"FFT   ------------ ------------ ------------ ------------                    "
"SIZE  PHASE  RATE   PHASE  RATE   PHASE  RATE   PHASE  RATE                  "
"----  -----  -----  -----  -----  -----  -----  -----  -----                 "
"                nco*1          nco*1          nco*1          nco*1          "
"<=512 0      fstc2  0      fstc2  0      fstc3  0      fstc3                 "
"                ^              ^              ^              ^              "
"                |______________|___           |______________|__            "
"                                |                             |              "
"                                |                             |              "
"                           one storage                  second storage       "
"                         element  (sign0)             element  (sign1)      "
"                                                                              "
"              nco*2   nco*1   nco*2          nco*2   nco*1   nco*2          "
"1K    0      fstc4  fstc0  fstc4  0      fstc5  fstc1  fstc5                 "
"                ^       ^       ^              ^       ^       ^            "
"                |_______|_______|___           |_______|_______|___         "
"                                    |                             |         "
"                                    |                             |         "
"                               one storage                  second storage   "
"                             element (sign0)              element   (sign1)  "
"                                                                              "
"              nco*2   nco*1   nco*2   nco*512 nco*2   nco*513 nco*2          "
"2K    0      fstc4    0    fstc4  (fstc6)      fstc4  (fstc6)      fstc4     "
"                ^              ^              ^              ^              "
"                |______________|______________|______________|___           "
"                                                                |             "
"                                                                |             "
"                                                          one storage         "
"                                                          element(sign0)      "
"                                                          (or sign1)          "
"                                                  (sign0 = sign1 for 2K)      "
"In the <=512 case, each delay center needs one storage element.               "
"In the 1K case, each delay center needs one storage element.                  "
"In the 2K case, only one storage element is required.                         "
"We do not need sign extension for fstc6 since it is obtained by"
"shifting the value to get the multiplication by 512.  This shifting"
"moves sign bits where they are needed."
      end PAL_7F
```

This page intentionally left balnk.

FFT Card PALASM Files

```
TITLE  U127 COUNTER CHIP MS BITS
PATTERN  2
REVISION B
AUTHOR JOE GREENBERG
COMPANY NRAO
DATE April 7, 1989
; in /vlbsoft/pal_prom/pals/ctrmsb.pds
; SEE L002D34.SCH FOR FUNCTIONAL DESCRIPTION.
; USED ON L002D05.SCH
chip counter_lsb pal16r8
;pins  1      2      3      4      5
       CLK    /RC    RAMCLKP       LDRAMPP       /WEPP
;pins  6      7      8      9      10
       /RST   NC     NC     NC     GND
;pins  11     12     13     14     15
       DISABLE       Q9     Q8     Q7     Q6
;pins  16     17     18     19      20
       /WEP   /LDRAMPPP CEP CL     VCC

EQUATIONS

; USE FOR 7 NS SPEED.
; LOGIC FOR PAL 10 BIT COUNTER
; IN 2 BIT SECTIONS.
;  2ND PAL   16R8
;INPUTS:
;      /WEPP (OUTPUT /WEP PUT THRU EXTERNAL FLIP FLOP.)
;      LDRAMPP
;      RAMCLKP
;      /RST
;       /RC    (FROM OTHER PAL)
;OUTPUTS:
LDRAMPPP := LDRAMPP
;WEP := ( RAMCLKP * LDRAMPP )\
WEP := RAMCLKP * LDRAMPP
; CL := RST + LDRAMPP*/LDRAMPPP (TO OTHER PAL)
/CL := /( RST*/LDRAMPP + LDRAMPP*/LDRAMPPP )
;/CEP := (WEP\*WEPP)\ * LDRAMPP (TO OTHER PAL)
;/CEP := (WEP + /WEPP) * LDRAMPP
/CEP := WEP*LDRAMPP + /WEPP*LDRAMPP
/Q6 := /( RC*/Q6*/CL + /RC*Q6*/CL )
; /Q7 := /( (Q6*RC)*/Q7*/CL + (Q6*RC)\*Q7*/CL )
; /Q7 := /( Q6*RC*/Q7*/CL + (/Q6+/RC)*Q7*/CL )
/Q7 := /( Q6*RC*/Q7*/CL + /Q6*Q7*/CL + /RC*Q7*/CL )
; /Q8 := /( (Q6*Q7*RC)*/Q8*/CL + (Q6*Q7*RC)\*Q8*/CL )
; /Q8 := /( Q6*Q7*RC*/Q8*/CL + (/Q6+/Q7+/RC)*Q8*/CL )
/Q8 := /( Q6*Q7*RC*/Q8*/CL + /Q6*Q8*/CL + /Q7*Q8*/CL + /RC*Q8*/CL )
; /Q9 := /( (Q6*Q7*Q8*RC)*/Q9*/CL + (Q6*Q7*Q8*RC)\*Q9*/CL  )
; /Q9 := /( Q6*Q7*Q8*RC*/Q9*/CL + (/Q6+/Q7+/Q8+/RC)*Q9*/CL  )
/Q9 := /( Q6*Q7*Q8*RC*/Q9*/CL + /Q6*Q9*/CL + /Q7*Q9*/CL + /Q8*Q9*/CL + /RC*Q9*/CL )
;WEPP := WEP FOR CE SIMULATION.  PIN5 = NC, PIN12=/WEPP
```

```
FFT CARD PAL
TITLE  74F153 & COUNTER LS BITS
PATTERN  1
REVISION B
AUTHOR JOE GREENBERG
COMPANY NRAO
DATE Apri1 3, 1989
; in /v1bsoft/pal_prom/pals/ctrlsb.pds
chip counter_lsb pal16r6
; PINS 2,3,4,5, & 6 EMULATE A 74F153 MUX WITH C2 &C3 TIED HIGH.
;pins  1      2      3      4      5
       CLK    C0     C1     A      B
;pins  6      7      8      9      10
       G      NC     CL     CE     GND
;pins  11     12     13     14     15
       DISABLE       /RC    Q5     Q4     Q3
;pins  16     17     18     19     20
       Q2     Q1     Q0     Y      VCC

EQUATIONS
; USE FOR 7 NS SPEED.
; LOGIC FOR PAL 10 BIT COUNTER
; IN 2 PAL SECTIONS.
; 1ST PAL   16R6
; INPUTS:
;      CL = CLEAR
;      CLK
;      CE = COUNT ENABLE
; DISABLE = TRI STATE OUTPUT DISABLE
; OUTPUTS:
; NOTE /Q := () CORRESPONDS TO Q := /()
;
;  74153 EMULATION WITH C2 & C3 TIED HIGH.
;/Y = /(G*/B*/A*C0 + G*/B*A*C1 + G*B*/A + G*B*A)
/Y = /(G*/B*/A*C0 + G*/B*A*C1 + G*B)
; LS BYTE OF COUNTER
/Q0 := /( CE*/Q0*/CL + /CE*Q0*/CL )
;/Q1 := /( (Q0*CE)*/Q1*/CL + (Q0*CE)\*Q1*/CL )
; /Q1 := /( Q0*CE*/Q1*/CL + (/Q0+/CE)*Q1*/CL )
/Q1 := /( Q0*CE*/Q1*/CL + /Q0*Q1*/CL + /CE*Q1*/CL )
; /Q2 := /( (Q0*Q1*CE)*/Q2*/CL + (Q0*Q1*CE)\*Q2*/CL )
; /Q2 := /( (Q0*Q1*CE)*/Q2*/CL + (/Q0+/Q1+/CE)*Q2*/CL )
/Q2 := /( Q0*Q1*CE*/Q2*/CL + /Q0*Q2*/CL + /Q1*Q2*/CL + /CE*Q2*/CL )
; /Q3 := /( (Q0*Q1*Q2*CE)*/Q3*/CL + (Q0*Q1*Q2*CE)\*Q3*/CL  )
; /Q3 := /( Q0*Q1*Q2*CE*/Q3*/CL + (/Q0+/Q1+/Q2+/CE)*Q3*/CL  )
/Q3 := /( Q0*Q1*Q2*CE*/Q3*/CL + /Q0*Q3*/CL + /Q1*Q3*/CL + /Q2*Q3*/CL + /CE*Q3*/CL  )
; /Q4 := /( (Q0*Q1*Q2*Q3*CE)*/Q4*/CL + (Q0*Q1*Q2*Q3*CE)\*Q4*/CL  )
; /Q4 := /( (Q0*Q1*Q2*Q3*CE)*/Q4*/CL + (/Q0+/Q1+/Q2+/Q3+/CE)*Q4*/CL  )
/Q4 := /( Q0*Q1*Q2*Q3*CE*/Q4*/CL + /Q0*Q4*/CL + /Q1*Q4*/CL + /Q2*Q4*/CL + /Q3*Q4*/CL
+ /CE*Q4*/CL  )
; /Q5 := /( (Q0*Q1*Q2*Q3*Q4*CE)*/Q5*/CL + (Q0*Q1*Q2*Q3*Q4*CE)\*Q5*/CL  )
; /Q5 := /( Q0*Q1*Q2*Q3*Q4*CE*/Q5*/CL + (/Q0+/Q1+/Q2+/Q3+/Q4+/CE)*Q5*/CL  )
/Q5 := /( Q0*Q1*Q2*Q3*Q4*CE*/Q5*/CL + /Q0*Q5*/CL + /Q1*Q5*/CL + /Q2*Q5*/CL
             + /Q3*Q5*/CL +/Q4*Q5*/CL + /CE*Q5*/CL  )
RC = Q0*Q1*Q2*Q3*Q4*Q5*CE
```

APPENDIX III FCC Memory Allocation:

THE FOLLOWING MEMORY LOCATION ASSIGNMENTS ARE FOR THE FFT CONTROL CARD

87C520 Microprocessor Memory Word Locations

LOC	FUNCTION	
00	REGISTER 0, BANK 0	
01	REGISTER 1, BANK 0	
02	REGISTER 2, BANK 0	
03	REGISTER 3, BANK 0	
04	REGISTER 4, BANK 0	
05	REGISTER 5, BANK 0	
06	REGISTER 6, BANK 0	
07	REGISTER 7, BANK 0	
08	REGISTER 0, BANK 1	
09	REGISTER 1, BANK 1	
0A	REGISTER 2, BANK 1	
0B	REGISTER 3, BANK 1	
0C	REGISTER 4, BANK 1	
0D	REGISTER 5, BANK 1	
0E	REGISTER 6, BANK 1	
0F	REGISTER 7, BANK 1	
10	REGISTER 0, BANK 2	USED IN NEW FRINGE CYCLE INTERRUPT
11	REGISTER 1, BANK 2	USED IN NEW FRINGE CYCLE INTERRUPT
12	REGISTER 2, BANK 2	USED IN NEW FRINGE CYCLE INTERRUPT
13	REGISTER 3, BANK 2	USED IN NEW FRINGE CYCLE INTERRUPT
14	REGISTER 4, BANK 2	USED IN NEW FRINGE CYCLE INTERRUPT
15	REGISTER 5, BANK 2	USED IN NEW FRINGE CYCLE INTERRUPT
16	REGISTER 6, BANK 2	USED IN NEW FRINGE CYCLE INTERRUPT
17	REGISTER 7, BANK 2	USED IN NEW FRINGE CYCLE INTERRUPT
18	REGISTER 0, BANK 3	USED IN HCB INTERRUPT
19	REGISTER 1, BANK 3	USED IN HCB INTERRUPT
1A	REGISTER 2, BANK 3	USED IN HCB INTERRUPT
1B	REGISTER 3, BANK 3	USED IN HCB INTERRUPT
1C	REGISTER 4, BANK 3	USED IN HCB INTERRUPT
1D	REGISTER 5, BANK 3	USED IN HCB INTERRUPT
1E	REGISTER 6, BANK 3	USED IN HCB INTERRUPT
1F	REGISTER 7, BANK 3	USED IN HCB INTERRUPT
20	BIT ADDRESSABLE	(BITS 00 THRU 07) USED IN MONITOR
21	BIT ADDRESSABLE	(BITS 08 THRU 0F) USED IN TEST
22	BIT ADDRESSABLE	(BITS 10 THRU 17) MISCL FLAGS
23	BIT ADDRESSABLE	(BITS 18 THRU 1F)
24	BIT ADDRESSABLE	(BITS 20 THRU 27)
25	BIT ADDRESSABLE	(BITS 28 THRU 2F)
26	BIT ADDRESSABLE	(BITS 30 THRU 37)
27	BIT ADDRESSABLE	(BITS 38 THRU 3F)
28	BIT ADDRESSABLE	(BITS 40 THRU 47)
29	BIT ADDRESSABLE	(BITS 48 THRU 4F)
2A	BIT ADDRESSABLE	(BITS 50 THRU 57)
2B	BIT ADDRESSABLE	(BITS 58 THRU 5F)
2C	BIT ADDRESSABLE	(BITS 60 THRU 67)
2D	BIT ADDRESSABLE	(BITS 68 THRU 6F)
2E	BIT ADDRESSABLE	(BITS 70 THRU 77)
2F	BIT ADDRESSABLE	(BITS 78 THRU 7F) USED IN TEST
30	START OF STACK	
31		
32		

```
56
57
58      STORAGE FOR MODEL TIME CODE
59      STORAGE FOR MODEL TIME CODE
5A      STORAGE FOR MODEL TIME CODE
5B      STORAGE FOR MODEL TIME CODE
5C      STORAGE FOR MODEL TIME CODE
5D      STORAGE FOR MODEL TIME CODE
5E      STORAGE FOR MODEL TIME CODE
5F      STORAGE FOR MODEL TIME CODE
60      LAST SENT 8200 BYTE
61      LAST SENT 8300 BYTE
62      LAST SENT 8500 BYTE
63      FRINGE CYCLE COUNTER
64
65      USED IN MONITOR
66      INT0 INTERRUPT VECTOR
67      INT0 INTERRUPT VECTOR
68      TIM0 INTERRUPT VECTOR
69      TIM0 INTERRUPT VECTOR
6A      INT1 INTERRUPT VECTOR
6B      INT1 INTERRUPT VECTOR
6C      TIM1 INTERRUPT VECTOR
6D      TIM1 INTERRUPT VECTOR
6E      UART INTERRUPT VECTOR
6F      UART INTERRUPT VECTOR
70      TEMP STORAGE IN RAMHCB
71      TEMP STORAGE IN RAMHCB
72      TEMP STORAGE IN RAMHCB
73      TEMP STORAGE IN RAMHCB
74      TEMP STORAGE IN RAMHCB
75      TEMP STORAGE IN RAMHCB
76      TEMP STORAGE IN RAMHCB
77      LAST TERMINAL KEY
78      MONITOR ADDRESS SHIFT IN
79      MONITOR ADDRESS SHIFT IN
7A      MONITOR ADDRESS SHIFT IN
7B      MONITOR ADDRESS SHIFT IN
7C      MONITOR SHIFT IN INDEX
7D      LAST MONITOR FUNCTION
7E      MONITOR TEMP STORAGE
7F      MONITOR TEMP STORAGE


        THE FOLLOWING BIT LOCATION ASSIGNMENTS ARE FOR THE FFT CONTROL CARD
```

87C520 Microprocessor Memory Bit Locations

BIT	FUNCTION	
00	MONITOR FUNCTIONS	DISPLAY,PAGE, TEST FLAG 1
01	MONITOR FUNCTIONS	NIBBLE FLAG
02	MONITOR FUNCTIONS	DISPLAY,PAGE, TEST FLAG 2
03	MONITOR FUNCTIONS	DISPLAY/PAGE INTERNAL,EXTERNAL MEM FLAG
04	MONITOR FUNCTIONS	D ONLY AND P ONLY FLAG
05	MONITOR FUNCTIONS	MODIFY FLAG
06	MONITOR FUNCTIONS	PARAMETER MEMORY FLAG
07	MONITOR FUNCTIONS	HELP FLAG
08	TEST FUNCTIONS	ALSO C/R ONLY FLAG
09	TEST FUNCTIONS	

```
0A      TEST FUNCTIONS
0B      TEST FUNCTIONS
0C      TEST FUNCTIONS
0D      TEST FUNCTIONS
0E      TEST FUNCTIONS
0F      TEST FUNCTIONS
10      HCB FLAG
11      GOOD XILINX LOAD FLAG
12      START OBS FLAG
13      USED IN RAMHCB MODEL ROUTINE (MARKS STATION 2 OR 7)
14      CLEAR MODEL TIME COUNTER FLAG
15      XON/XOFF FLAG
16      TEMP FLAG (USED IN RAMHCB)
17      OBS FLAG
18      SELF TEST FLAG #1
19      SELF TEST FLAG #2
1A      SELF TEST FLAG #3
1B       MODEL SEND FLAG
1C
.
.
.
77
78      USED IN TEST
79      USED IN TEST
7A      USED IN TEST
7B      USED IN TEST
7C      USED IN TEST
7D      USED IN TEST
7E      USED IN TEST
7F      USED IN TEST
```

Storage Assignments in FCC Main Memory;

```
0000 TO 021F TEMP STORAGE FOR MODEL PARAMETERS (SEE TEMPMOD IN RAMHCB.ASM)
0220 TO 0221 PULSAR CHECKSUM.  SEE PCKSUM IN RAMHCB.ASM.
0300 TO 0358 TEMP STORAGE FOR MODEL PARAMETERS (SEE TEMPSTO IN MONITOR.ASM)
1000 TO 4FFF PROGRAM MEMORY.  SEE FCCALL.MAP.
6000 TO 6A14 TEMP STORAGE FOR FFT CARD RAM BYTES

7D00  ACTIVE MODEL FOR STATION 0
7D01  ACTIVE MODEL FOR STATION 1
7D02  ACTIVE MODEL FOR STATION 2
7D03  ACTIVE MODEL FOR STATION 3
7D04  ACTIVE MODEL FOR STATION 4
7D05  ACTIVE MODEL FOR STATION 5
7D06  ACTIVE MODEL FOR STATION 6
7D07  ACTIVE MODEL FOR STATION 7
7D08  ACTIVE MODEL FOR STATION 8
7D09  ACTIVE MODEL FOR STATION 9
7D0A  ACTIVE PULSAR MODEL

7D10  NEXT MODEL FOR STATION 0
7D11  NEXT MODEL FOR STATION 1
7D12  NEXT MODEL FOR STATION 2
7D13  NEXT MODEL FOR STATION 3
7D14  NEXT MODEL FOR STATION 4
7D15  NEXT MODEL FOR STATION 5
7D16  NEXT MODEL FOR STATION 6
```

```
7D17  NEXT MODEL FOR STATION 7
7D18  NEXT MODEL FOR STATION 8
7D19  NEXT MODEL FOR STATION 9
7D1A  NEXT PULSAR MODEL
```

Memory Allocation for the Parameter Memory RAM;

ADDRESS	FUNCTION
0000	MISCELLANEOUS
0100	PULSAR MODEL #0
0118	PULSAR MODEL #1
0130	PULSAR MODEL #2
0148	PULSAR MODEL #3
0160	TIME MODEL #0 (22 Values)
0180	TIME MODEL #1
01A0	TIME MODEL #2
01C0	TIME MODEL #3
0200	SELF TEST DELAY MODEL
0300	SELF TEST FRINGE MODEL
0800	DELAY MODEL #0
0A00	DELAY MODEL #1
0C00	DELAY MODEL #2
0E00	DELAY MODEL #3
1000	FRINGE MODEL #0
1400	FRINGE MODEL #1
1800	FRINGE MODEL #2
1C00	FRINGE MODEL #3

FCC TIME VALUES:

MODEL FOR	MODEL 0	MODEL 1	MODEL 2	MODEL 3
STATION 0	160	180	1A0	1C0
STATION 1	162	182	1A2	1C2
STATION 2	164	184	1A4	1C4
STATION 3	166	186	1A6	1C6
STATION 4	168	188	1A8	1C8
STATION 5	16A	18A	1AA	1CA
STATION 6	16C	18C	1AC	1CC
STATION 7	16E	18E	1AE	1CE
STATION 8	170	190	1B0	1D0
STATION 9	172	192	1B2	1D2
PULSAR	174	194	1B4	1D4

FCC DELAY MODELS:

MODEL FOR	MODEL 0	MODEL 1	MODEL 2	MODEL 3
SELF TEST, DLYCENTER 0	0200			
SELF TEST, DLYCENTER 1	0218			
STATION 0, DLYCENTER 0	0800	0A00	0C00	0E00
STATION 0, DLYCENTER 1	0818	0A18	0C18	0E18
STATION 1, DLYCENTER 0	0830	0A30	0C30	0E30
STATION 1, DLYCENTER 1	0848	0A48	0C48	0E48

STATION 2, DLYCENTER 0	0860	0A60	0C60	0E60
STATION 2, DLYCENTER 1	0878	0A78	0C78	0E78
STATION 3, DLYCENTER 0	0890	0A90	0C90	0E90
STATION 3, DLYCENTER 1	08A8	0AA8	0CA8	0EA8
STATION 4, DLYCENTER 0	08C0	0AC0	0CC0	0EC0
STATION 4, DLYCENTER 1	08D8	0AD8	0CD8	0ED8
Note F0 to FF is not used				
STATION 5, DLYCENTER 0	0900	0B00	0D00	0F00
STATION 5, DLYCENTER 1	0918	0B18	0D18	0F18
STATION 6, DLYCENTER 0	0930	0B30	0D30	0F30
STATION 6, DLYCENTER 1	0948	0B48	0D48	0F48
STATION 7, DLYCENTER 0	0960	0B60	0D60	0F60
STATION 7, DLYCENTER 1	0978	0B78	0D78	0F78
STATION 8, DLYCENTER 0	0990	0B90	0D90	0F90
STATION 8, DLYCENTER 1	09A8	0BA8	0DA8	0FA8
STATION 9, DLYCENTER 0	09C0	0BC0	0DC0	0FC0
STATION 9, DLYCENTER 1	09D8	0BD8	0DD8	0FD8
Note F0 to FF is not used				

FCC FRINGE MODELS:

MODEL FOR	MODEL 0	MODEL 1	MODEL 2	MODEL 3
SELF TEST, CHANNEL 3	0300			
SELF TEST, CHANNEL 2	0318			
SELF TEST, CHANNEL 1	0330			
SELF TEST, CHANNEL 0	0348			
STATION 0, CHANNEL 3	1000	1400	1800	1C00
STATION 0, CHANNEL 2	1018	1418	1818	1C18
STATION 0, CHANNEL 1	1030	1430	1830	1C30
STATION 0, CHANNEL 0	1048	1448	1848	1C48
STATION 1, CHANNEL 3	1060	1460	1860	1C60
STATION 1, CHANNEL 2	1078	1478	1878	1C78
STATION 1, CHANNEL 1	1090	1490	1890	1C90
STATION 1, CHANNEL 0	10A8	14A8	18A8	1CA8
STATION 2, CHANNEL 3	10C0	14C0	18C0	1CC0
STATION 2, CHANNEL 2	10D8	14D8	18D8	1CD8
Note F0 to FF is not used				
STATION 2, CHANNEL 1	1100	1500	1900	1D00
STATION 2, CHANNEL 0	1118	1518	1918	1D18
STATION 3, CHANNEL 3	1130	1530	1930	1D30
STATION 3, CHANNEL 2	1148	1548	1948	1D48
STATION 3, CHANNEL 1	1160	1560	1960	1D60
STATION 3, CHANNEL 0	1178	1578	1978	1D78
STATION 4, CHANNEL 3	1190	1590	1990	1D90
STATION 4, CHANNEL 2	11A8	15A8	19A8	1DA8
STATION 4, CHANNEL 1	11C0	15C0	19C0	1DC0
STATION 4, CHANNEL 0	11D8	15D8	19D8	1DD8
Note F0 to FF is not used				
STATION 5, CHANNEL 3	1200	1600	1A00	1E00
STATION 5, CHANNEL 2	1218	1618	1A18	1E18
STATION 5, CHANNEL 1	1230	1630	1A30	1E30
STATION 5, CHANNEL 0	1248	1648	1A48	1E48

STATION	6, CHANNEL 3	1260	1660	1A60	1E60
STATION	6, CHANNEL 2	1278	1678	1A78	1E78
STATION	6, CHANNEL 1	1290	1690	1A90	1E90
STATION	6, CHANNEL 0	12A8	16A8	1AA8	1EA8
STATION	7, CHANNEL 3	12C0	16C0	1AC0	1EC0
STATION	7, CHANNEL 2	12D8	16D8	1AD8	1ED8
	Note F0 to FF is not used				
STATION	7, CHANNEL 1	1300	1700	1B00	1F00
STATION	7, CHANNEL 0	1318	1718	1B18	1F18
STATION	8, CHANNEL 3	1330	1730	1B30	1F30
STATION	8, CHANNEL 2	1348	1748	1B48	1F48
STATION	8, CHANNEL 1	1360	1760	1B60	1F60
STATION	8, CHANNEL 0	1378	1778	1B78	1F78
STATION	9, CHANNEL 3	1390	1790	1B90	1F90
STATION	9, CHANNEL 2	13A8	17A8	1BA8	1FA8
STATION	9, CHANNEL 1	13C0	17C0	1BC0	1FC0
STATION	9, CHANNEL 0	13D8	17D8	1BD8	1FD8
	Note F0 to FF is not used				

THE DELAY MODEL PARAMETER FORMAT IS SEEN BELOW;

ADDRESS	FUNCTION	
0000	P5	5TH ORDER PHASE POLYNOMIAL COEFFICIENT
0002	P4	4TH ORDER PHASE POLYNOMIAL COEFFICIENT
0004	P3	3RD ORDER PHASE POLYNOMIAL COEFFICIENT
0006	P2	2ND ORDER PHASE POLYNOMIAL COEFFICIENT
0008	P1	1ST ORDER PHASE POLYNOMIAL COEFFICIENT
000A	P0	0TH ORDER PHASE POLYNOMIAL COEFFICIENT
000C	M0	MODE DEPENDENT SHIFT PARAMETER *
000E	R4	4TH ORDER RATE POLYNOMIAL COEFFICIENT
0010	R3	3RD ORDER RATE POLYNOMIAL COEFFICIENT
0012	R2	2ND ORDER RATE POLYNOMIAL COEFFICIENT
0014	R1	1ST ORDER RATE POLYNOMIAL COEFFICIENT
0016	R0	0TH ORDER RATE POLYNOMIAL COEFFICIENT

*** SHIFT PARAMETER FORMAT IS SEEN BELOW;**

TRANSFORM	M0 (DOUBLE PRECISION INTEGER)
64 POINT	0000 0000 0000 0000
128 POINT	0000 0000 0000 0001
256 POINT	0000 0000 0000 0002
512 POINT	0000 0000 0000 0003
1024 POINT	0000 0000 0000 0004
2048 POINT	0000 0000 0000 0005

THE FRINGE/PULSAR MODEL PARAMETER FORMAT IS SEEN BELOW;

ADDRESS	FUNCTION	
0000	P5	5TH ORDER PHASE POLYNOMIAL COEFFICIENT
0002	P4	4TH ORDER PHASE POLYNOMIAL COEFFICIENT
0004	P3	3RD ORDER PHASE POLYNOMIAL COEFFICIENT
0006	P2	2ND ORDER PHASE POLYNOMIAL COEFFICIENT
0008	P1	1ST ORDER PHASE POLYNOMIAL COEFFICIENT
000A	P0	0TH ORDER PHASE POLYNOMIAL COEFFICIENT
000C	R4	4TH ORDER RATE POLYNOMIAL COEFFICIENT
000E	R3	3RD ORDER RATE POLYNOMIAL COEFFICIENT
0010	R2	2ND ORDER RATE POLYNOMIAL COEFFICIENT
0012	R1	1ST ORDER RATE POLYNOMIAL COEFFICIENT
0014	R0	0TH ORDER RATE POLYNOMIAL COEFFICIENT
0016	NOT USED	

This page intentionally left balnk.

APPENDIX IV VLBA1 ASIC Pin List

NAME	PIN	FINGER	PAD	PAD2	PAD3	U#	PART#	FFTSTG0	STG 1-5	MACNAME
AUXIN0	D02	7	12			26	TLCHTD	AUXIN0	AUXIN0	X
AUXIN1	C02	5	9			27	TLCHTD	AUXIN1	AUXIN1	X
AUXIN10	A09	101	196			18	TLCHTD	AUXIN10	AUXIN10	X
AUXIN11	A10	99	192			19	TLCHTD	AUXIN11	AUXIN11	X
AUXIN12	B10	97	186			20	TLCHTD	AUXIN12	AUXIN12	X
AUXIN13	B11	95	182			21	TLCHTD	AUXIN13	AUXIN13	X
AUXIN14	A12	93	179			22	TLCHTD	AUXIN14	AUXIN14	X
AUXIN15	C12	89	173			23	TLCHTD	AUXIN15	AUXIN15	X
AUXIN16	B13	87	170			24	TLCHTD	X	AUXIN16	X
AUXIN17	D12	85	166			25	TLCHTD	X	AUXIN17	X
AUXIN2	B01	3	6			28	TLCHTD	AUXIN2	AUXIN2	X
AUXIN3	C04	118	232			11	TLCHTD	AUXIN3	AUXIN3	X
AUXIN4	A03	116	230			12	TLCHTD	AUXIN4	AUXIN4	X
AUXIN5	C05	114	225			13	TLCHTD	AUXIN5	AUXIN5	X
AUXIN6	B05	112	221			14	TLCHTD	AUXIN6	AUXIN6	X
AUXIN7	C06	110	217			15	TLCHTD	AUXIN7	AUXIN7	X
AUXIN8	A06	108	213			16	TLCHTD	AUXIN8	AUXIN8	X
AUXIN9	B08	103	200			17	TLCHTD	AUXIN9	AUXIN9	X
CLK	H13	74	142			121	DRVT16U	CLK	CLK	CLK
CLKA	A07	107	209			89	TLCHTU	CLKA	CLKA	CLKA
CRBIT	N12	57	113			86	B4R	CRBIT	CRBIT	CRBIT
CRSFTCLK	L09	54	104			113	DRVT4U	CRSFTCLK	CRSFTCLK	CRSFTCLK
CRSI	N10	53	101			112	TLCHTD	CRSI	CRSI	CRSI
CRSO	N11	56	112			48	B4R	CRSO	CRSO	CRSO
CRSTB	M10	55	110			85	DRVT2U	CRSTB	CRSTB	CRSTB
DATAENBL_	N07	47	91			119	TLCHTD	PULSARG_	PULSARG_	CLRACCUM
EXPOF_	M09	52	100			50	BT8OD	EXPOF_	EXPOF_	X
EXTADD_0	E03	8	14			33	BD8TRPU	EXTADD_0	EA_0NCOT	EXTADD_0
EXTADD_1	D01	9	16			34	BD8TRPU	EXTADD_1	EA_1NCO0	EXTADD_1
EXTADD_2	E02	10	18			35	BD8TRPU	EXTADD_2	EA_2NCO1	EXTADD_2
EXTADD_3	E01	11	20			36	BD8TRPU	EXTADD_3	EA_3NCO2	EXTADD_3
EXTADD_4	F03	12	21			37	BD8TRPU	EXTADD_4	EA_4NCO3	EXTADD_4
EXTADD_5	F02	13	25			38	BD8TRPU	EXTADD_5	EA_5NCO4	EXTADD_5
EXTADD_6	F01	14	27			39	BD8TRPU	EXTADD_6	EA_6NCO5	EXTADD_6
EXTADD_7	G02	15	29			40	BD8TRPU	EXTADD_7	EA_7NCO6	EXTADD_7
EXTADD_8	H01	18	36			41	BD8TRPU	EXTADD_8	EA_8NCO7	EXTADD_8
EXTASEL	N08	48	92			117	TLCHTD	EXTASEL	EXTASEL	EXTSEL
EXTBSEL	M08	49	93			118	TLCHTD	EXTBSEL	EXTBSEL	COL_
FCLK	B07	This pin is not internally connected to anything.								
GND	C03	1	3					GND	GND	GND
GND	G03	16	30	32	31			GND	GND	GND
GND	N01	30	57					GND	GND	GND
GND	M07	45	89	87	88			GND	GND	GND
GND	M11	59	116					GND	GND	GND
GND	G12	75	147	145	146			GND	GND	GND
GND	A13	90	174					GND	GND	GND
GND	C07	106	205	207	206			GND	GND	GND
NCOCE_	M04	37	70			102	TLCHTD	NCOCE_	NCOCE_	MACWE_
NCOCI	L04	34	65			44	TLCHTU	NCOCI	NCOCI	TWID13
NCOCLK	J03	24	49			152	DRVT4U	NCOCLK	NCOCLK	X
NCOCO	M06	43	84			88	B2	NCOCO	NCOCO	X
NCODOUT9	H02	19	38			42	B8RP	X	NCO8	X
NCOLDIN_	N04	39	76			45	TLCHTU	NCOLDIN_	NCOLDIN_	X
NCOOE	M03	35	66			55	TLCHTU	NCOOE	NCOOE	NCOOE
NCORW_	L06	42	83			47	TLCHTU	NCORW_	NCORW_	X
NCOSI	N02	33	62			43	TLCHTU	NCOSI	NCOSI	TWID12
NCOSO	M05	40	79			87	B2	NCOSO	NCOSO	X
NCOSRCLK	L05	38	74			120	DRVT2U	NCOSRCLK	NCOSRCLK	X
NCOSTA_	N05	41	80			46	TLCHTU	NCOSTA_	NCOSTA_	AUXOUT
NDATAIN0	C01	6	10			29	TLCHTD	NDATAIN0	NDATAIN0	X
NDATAIN1	D03	4	8			30	TLCHTD	NDATAIN1	NDATAIN1	X

NRAO DRAWING NR A56000D001 VLBA1 ASIC PIN LIST LOTUS FILE D001L01.WK1

NAME	PIN	FINGER	PAD	PAD2	PAD3	U#	PART#	FFTSTG0	STG 1-5	MACNAME
NDATAIN10	C08	102	199			94	TLCHTD	NDATAIN10	NDATAIN10	NDATAIN10
NDATAIN11	B09	100	194			95	TLCHTD	NDATAIN11	NDATAIN11	NDATAIN11
NDATAIN12	C09	98	188			96	TLCHTD	NDATAIN12	NDATAIN12	NDATAIN12
NDATAIN13	A11	96	184			97	TLCHTD	NDATAIN13	NDATAIN13	NDATAIN13
NDATAIN14	C10	94	180			98	TLCHTD	NDATAIN14	NDATAIN14	NDATAIN14
NDATAIN15	B12	92	178			99	TLCHTD	NDATAIN15	NDATAIN15	NDATAIN15
NDATAIN16	D11	88	172			100	TLCHTD	X	NDATAIN16	NDATAIN16
NDATAIN17	C13	86	168			101	TLCHTD	X	NDATAIN17	NDATAIN17
NDATAIN2	B02	2	5			31	TLCHTD	NDATAIN2	NDATAIN2	X
NDATAIN3	B03	119	233			90	TLCHTD	NDATAIN3	NDATAIN3	NDATAIN3
NDATAIN4	A02	117	231			91	TLCHTD	NDATAIN4	NDATAIN4	NDATAIN4
NDATAIN5	B04	115	229			92	TLCHTD	NDATAIN5	NDATAIN5	NDATAIN5
NDATAIN6	A04	113	222			93	TLCHTD	NDATAIN6	NDATAIN6	NDATAIN6
NDATAIN7	A05	111	219			51	TLCHTD	NDATAIN7	NDATAIN7	X
NDATAIN8	B06	109	215			52	TLCHTD	NDATAIN8	NDATAIN8	X
NDATAIN9	A08	104	201			53	TLCHTD	NDATAIN9	NDATAIN9	X
OUT0	E11	84	164			57	BT4RP	OUT0	OUT0	OUT0
OUT1	D13	83	162			58	BT4RP	OUT1	OUT1	OUT1
OUT10	J11	68	131			67	BT4RP	OUT10	OUT10	OUT10
OUT11	K12	67	129			68	BT4RP	OUT11	OUT11	OUT11
OUT12	L13	66	127			69	BT4RP	OUT12	OUT12	OUT12
OUT13	L12	65	125			70	BT4RP	OUT13	OUT13	OUT13
OUT14	K11	64	123			71	BT4RP	OUT14	OUT14	OUT14
OUT15	M13	63	122			72	BT4RP	OUT15	OUT15	OUT15
OUT16	M12	62	121			73	BT4RP	OUT16	OUT16	OUT16
OUT17	L11	61	120			74	BT4RP	OUT17	OUT17	OUT17
OUT2	E12	82	160			59	BT4RP	OUT2	OUT2	OUT2
OUT3	E13	81	158			60	BT4RP	OUT3	OUT3	OUT3
OUT4	F11	80	156			61	BT4RP	OUT4	OUT4	OUT4
OUT5	F12	79	154			62	BT4RP	OUT5	OUT5	OUT5
OUT6	F13	78	152			63	BT4RP	OUT6	OUT6	OUT6
OUT7	G13	77	150			64	BT4RP	OUT7	OUT7	OUT7
OUT8	J12	70	135			65	BT4RP	OUT8	OUT8	OUT8
OUT9	K13	69	133			66	BT4RP	OUT9	OUT9	OUT9
PNAND	H11	72	139			54	B1	X	X	X
R2CNT0	L02	29	55			49	B8RP	R2CNT0	R2CNT0	X
RAMTEST_	L10	58	114			122	TLCHNU	X	X	X
RST_	J13	71	137			114	TLCHTU	RST_	RST_	RST_
TWID0	H03	20	40			32	BD2TD	TWID0	TWID0	TWID11
TWID1	J01	21	42			103	BD2TD	TWID1	TWID1	TWID1
TWID2	J02	22	45			104	BD2TD	TWID2	TWID2	TWID2
TWID3	K01	23	47			105	BD2TD	TWID3	TWID3	TWID3
TWID4	L07	46	90			106	TLCHTD	TWID4	TWID4	TWID4
TWID5	K02	25	51			107	BD2TD	TWID10	TWID5	TWID10
TWID6	L01	26	52			108	BD2TD	TWID11	TWID6	TWID6
TWID7	M01	27	53			109	BD2TD	TWID12	TWID7	TWID7
TWID8	K03	28	54			110	BD2TD	TWID13	TWID8	TWID8
TWID9	M02	32	61			111	TLCHTD	TWID9	TWID9	TWID9
TWIDOE	N03	36	67			56	TLCHTU	TWIDOE	TWIDOE	TWIDOE
VDD	A01	120	234	1				VDD	VDD	VDD
VDD	G01	17	34					VDD	VDD	VDD
VDD	L03	31	60	59				VDD	VDD	VDD
VDD	N06	44	85					VDD	VDD	VDD
VDD	N13	60	117	118				VDD	VDD	VDD
VDD	H12	73	141					VDD	VDD	VDD
VDD	G11	76	148					VDD	VDD	VDD
VDD	C11	91	177	176				VDD	VDD	VDD
WEIN_A	L08	50	94			115	TLCHTD	WEIN_A	WEIN_A	WEIN_
WEIN_B	N09	51	99			116	TLCHTD	WEIN_B	WEIN_B	ROW_

APPENDIX V HCB Protocol for the FFT Test Fixture

Stored in /home/azalea/corrdwgs/ffttfix/up/fftfix.hcb

SECTION A:

A brief summary of each command is listed in this section. See SECTION B for additional details of each command. Commands are grouped according to function, and are in numerical order by function code within a group.

The sum of the number of bytes to the target plus the number of bytes to be received back from the target must be greater than 1. Thus, some commands require a second dummy byte to follow the function code byte.

		Fnct Code hex	Additional Bytes
1)	MAC CONTROL WORDS		
a)	read test byte:	0E	(one byte returned)
b)	set mac byte:	12	bb
c)	send MAC control words:	15	AA DD----DD (128 bytes of data)
d)	send check sum of a):	16	(two bytes returned)
e)	strobe control words:	17	(second dummy byte sent)
f)	shift test byte:	25	aa dd
2)	FFT CONTROL WORDS		
a)	read test byte:	0D	(readback for FFT control word)
b)	rd/rst expovflow:	14	(one byte returned)
3)	FFT/MAC TESTS		
a)	snapshot:	08	DD (DD = 0, 1, 2 or 3 chipset nr)
b)	load data ram:	09	CC PP DD----DD (1024 bytes of data)
c)	start fft test:	0A	(one dummy byte returned at completion)
d)	bank switch:	0B	(one dummy byte returned at completion)
e)	read results:	0C	xx (xx specifies result # 0 thru 255)
f)	compsnap:	0F	DD-DD (32 bytes tell which VLBA1 to test)
g)	sendsnap	10	xx (target sends snapshot block # xx)
h)	stop sequencer:	13	(second dummy byte sent)
j)	rcvsnap:	18	xx DD--DD (target rcvs snapshot blk # xx)
k)	senderr:	19	(target sends 32 sixteen bit err counts)
l)	stopcomp:	1A	(stop compsnap test and send err counts)

SECTION B:

Additional details for each command in SECTION A are given in this section.

****

1) Details for MAC Control Words

a) FUNCTION CODE OE: read test byte

0E (one byte returned)

read test byte: reads the 8 bit register in the test fixture that contains the last byte shifted out of the tail end of the MAC control word chain;

b) FUNTION CODE 12: set mac byte

12 bb

writes an 8 bit control register in the test fixture

c) FUNCTION CODE 15: send MAC control words

15 AA DD----DD (128 bytes of data)

AA is a card assignment (00 to 0E in the system, 00 in test fixture) DD----DD indicates 128 bytes of data (four bytes per asic, bytes within an asic are sent LS byte first, LS bit first; words are sent in the daisy chain order, control word for U61 is sent first, control word for U107 is sent last)

d) FUNCTION CODE 16: send check sum for last MAC control words

16 (two bytes returned)

checksum is made as bytes are received from the HCB

e) FUNCTION CODE 17: strobe MAC control words

17 (second dummy byte sent)

strobes the MAC control words into primary storage

f) FUNCTION CODE 25: shift test byte

25 aa dd

the shift test takes the byte dd and shifts it to card aa (aa=00 in test fixture)

****

2) Details for FFT Control Words

a) FUNCTION CODE 0D: read test byte

0D (readback for FFT control word)

read test byte: reads the 8 bit register in the test fixture that contains the last byte shifted OUT of the tail end of the FFT Card control word chain;

b) FUNCTION CODE 14: rd/rst expovflow

14 (one byte returned)

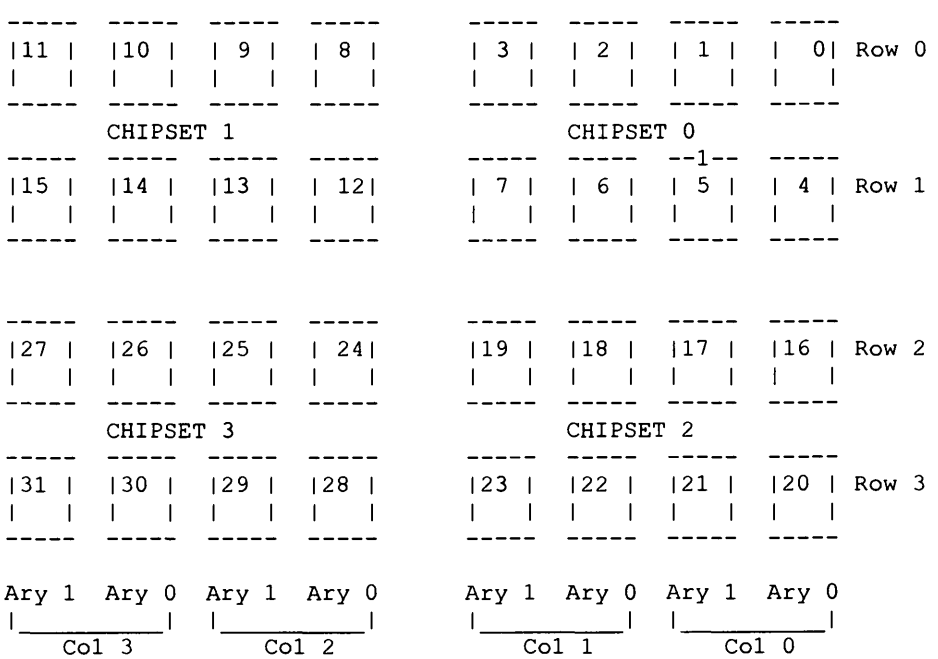
reads and resets the exponent overflow latch in the test fixture

3) Details for FFT/MAC tests

a) FUNCTION CODE 08: snapshot

08 DD (DD = 0, 1, 2 or 3 chipset nr)

Target takes a snapshot of the MAC results in one chipset, where a chipset consists of 8 VLBAls on the MAC card as follows: (as viewed from front of MAC card)

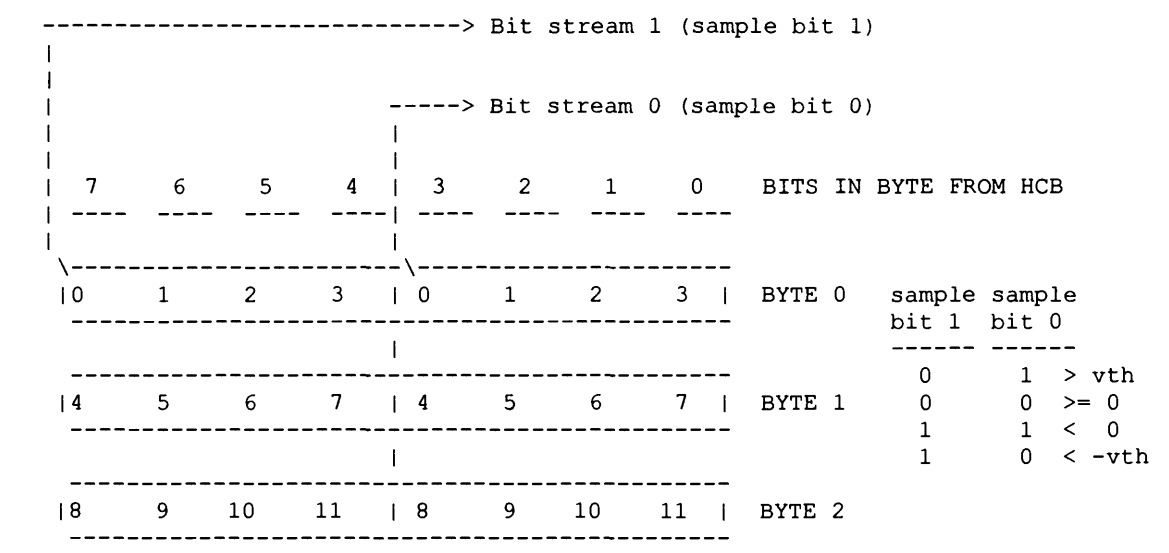


b) FUNCTION CODE 09: load data ram

09 CC PP DD----DD (1024 bytes of data)

Load the data ram in the test fixture:

CC is the channel number 0-3 (FFT pipeline 0-3)
PP is the page number 0-63 (each page is 1024 bytes)
DD----DD is 1024 bytes for one page



The data rams are 64K nibbles deep. A single byte on the HCB goes to two 64K by 4 bits wide rams. The four lower significant bits of the byte become part of bit stream 0 for the data channel. The four most significant bits become part of bit stream 1 for the data channel.

Each bit stream thus has a total of 65536 x 4 = 262,144 bits.

The order of shifting the bit stream is as indicated above, where the most significant bit of a nibble shifts out first, starting with bit 0 of the bit stream and going up through bit 262,143 which is the least significant bit of byte # 65535 for bit stream 0.

c) FUNCTION CODE 0A: start fft test

0A (one dummy byte returned at completion)

Start a test. A dummy byte is returned when the test fixture is ready to continue.

d) FUNCTION CODE 0B: bank switch

0B (one dummy byte returned at completion)

Bandswitch the MAC card and continue a test. A dummy byte is returned when data is ready to be read.

e) FUNCTION CODE 0C: read results

0C xx (xx specifies result # 0 through 255)

Read MAC results: xx = result nr 0 through 255

The data read back consists of 40,960 bytes, read back in blocks of 1280 bytes, 32 blocks total ordered as follows:

```
int totalcnt = 40960;   /* 5 bytes/result, 256 results/chip, 32 chips */
int blockcnt = 1280;    /* 5 bytes/result, 8 results/chip, 32 chips */
```

extadr	array	row	col	b7	b6	b5	b4	b3	b2	b1	b0	
base+0	0	0	0	e5	e5	e5	e4	e3	e2	e1	e0	2's comp exp
				i7	i6	i5	i4	i3	i2	i1	i0	1's comp imag
				i14	i14	i13	i12	i11	i10	i9	i8	mantissa
				r7	r6	r5	r4	r3	r2	r1	r0	1's comp real
				r14	r14	r13	r12	r11	r10	r9	r8	mantissa
base+0	0	0	1	e5	e5	e5	e4	e3	e2	e1	e0	2's comp exp
				i7	i6	i5	i4	i3	i2	i1	i0	1's comp imag
				i14	i14	i13	i12	i11	i10	i9	i8	mantissa
				r7	r6	r5	r4	r3	r2	r1	r0	1's comp real
				r14	r14	r13	r12	r11	r10	r9	r8	mantissa

For each block of 1280 results, the results are in the format above. The col, row, array and extadr are cycled thru in that order (col is LS, extadr is MS). For block 0, the extadr base is zero, for block 1 it is 8 etc. Thus a block contains 8 accumulation results from each of 32 VLBA1 chips, and a total of 32 blocks are read to get all 256 results.

f) FUNCTION CODE 0F: compsnap

0F DD-DD (32 bytes tell which VLBA1 to test)

Requests that a continuous test of MAC results be done, comparing each 131 ms set of results to the snapshot. The 32 byte data buffer has one byte for each VLBA1 chip on the MAC card. If the byte is zero, the results for the VLBA1 are not to be tested. The order of the bytes 0-31 relates directly to the chip index in the diagram for FUNCTION CODE 15. The error counters (see FUNCTION CODE 19) are zeroed.

g) FUNCTION CODE 10: sendsnap

10 xx (target sends snapshot block # xx)

The complete snapshot is 10240 bytes, transferred in 10 blocks of 1024 bytes each. xx is 0-9.

The last snapshot that was taken is sent for storage on disk. May eventually want to know the data format for analysis, but for now the only purpose is storage.

h) FUNCTION CODE 13: stop sequencer

13 (second dummy byte sent)

Stop the test fixture sequencer in preparation for starting a test.

j) FUNCTION CODE 18: rcvsnap

18 xx DD--DD (target rcvs snapshot block # xx)

The complete snapshot is 10240 bytes, transferred in 10 blocks of 1024 bytes each. xx is 0-9.

A snapshot that has been stored on disk is sent to the target for use.

k) FUNCTION CODE 19: senderr

19 (target sends 32 sixteen bit err counts)

The current state of the error counters is sent by the target. There is a 16 bit counter for each VLBA1 on the MAC card. A counter is incremented each time a MAC result for a chip does not compare with the snapshot. The order of 16 bit values 0-31 relates directly to the chip index in the diagram for FUNCTION CODE 15. The 16 bit values are sent MS byte first.

l) FUNCTION CODE 1A: stopcomp

1A (stop compsnap test and send err counts)

Stop the continuous test that was started with FUCNTION CODE 16. Also send the error counts (see FUNCTION CODE 19).

****

CHECKSUM IS OF THE FORM:

```
AAAAAAAA AAAAAAAA   ACCUMULATED SUM
+00000000 BBBBBBBB  CURRENT BYTE BEING SUMMED INTO CHECKSUM
-----------------
AAAAAAAA AAAAAAAA   NEW ACCUMULATOR SUM
```

The transfer of the checksum back over the HCB is done MS byte first.

****

Note: There is a requirement that the total number of bytes to and from a target must be greater than 1. Thus if a single byte function code is sent to a target and no response is expected, the transfer to the target will include a second byte. The contents of this dummy byte are not specified.

APPENDIX VI AM29C327 Floating-Point Processor Chip

This section contains a copy of the AM29C327 Data Sheet. Note AMD no longer manufactures this component.

This page intentionally left blank.

Am29C327

Double-Precision Floating-Point Processor

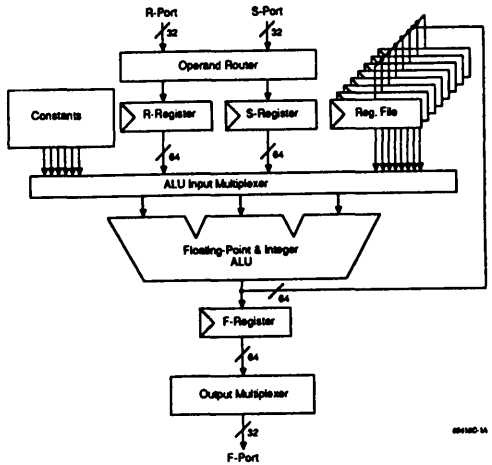
Am29C327

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- High-performance double-precision floating-point processor
- Comprehensive floating-point and integer instruction sets
- Single VLSI device performs single-, double-, and mixed-precision operations
- Performs conversions between precisions and between data formats
- Complies with seven industry-standard floating-point formats
  - ANSI/IEEE standard 754-1985; single- and double-precision
  - DEC F, DEC D, DEC G
  - IBM single- and double-precision
- Exact IEEE compliance for denormalized numbers with no speed penalty
- Eight-deep register file for intermediate results and on-chip 64-bit data path facilitates compound operations; e.g., Newton-Raphson division, sum-of-products, and transcendentals
- Supports pipelined or flow-through operation
- Fabricated with Advanced Micro Devices' 1.2 micron CMOS process

SIMPLIFIED BLOCK DIAGRAM



BD006080

Publication #	Rev.	Amendment
09418	C	/0

Issue Date: December 1988

GENERAL DESCRIPTION

The Am29C327 double-precision floating-point processor is a single VLSI device that implements an extensive floating-point and intager instruction set. The three most widely used floating-point standards – IEEE, DEC, and IBM – are supported for both single- and double-precision operations. IEEE operations comply with the ANSI/IEEE Standard 754, with direct implamentation of special features such as gradual underflow and handling of traps and denormalized numbers.

The Am29C327 consists of a 64-bit ALU, a 64-bit datapath, and a control unit. The ALU hes three data input ports, and can perform single-operand, two-operand, and three-operand operations. The data path comprises two 64-bit input operand registers, an 8-by-64-bit register file for storage of intermediate results, thraa operand-selection multiplexers that provide for orthogonal selection of input operands, a 64-bit output register, and an output multiplexer that allows access to the 32 MSBs or 32 LSBs of the result data. Control signals determine the operation to be performed, the sourca of operands, the operand precisions, the rounding mode, and other aspects of device operation.

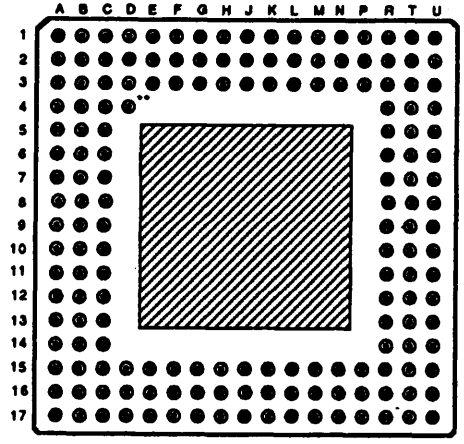
Operations can be performed in either of two modas: flow-through or pipelined. In the flow-throug!; mode, the ALU is completely combinatorial; this mode is best suited for scalar operations. Pipelined mode divides the ALU into one or two pipelina stages, for use in vector operations, as often found in graphics or signal processing.

Fabricated with AMD's 1.2 *micron* CMOS technology, the Am29C327 is housed In a 169-laad pin-grid-array (PGA) package.

RELATED AMD PRODUCTS

Part No.	Description
Am29C10A	CMOS 12-Bit Sequencer
Am29C111	CMOS 18-Bit Sequencer
Am29C116	CMOS 16-Bit Microprocessor
Am29CPL141	CMOS 64 x 16 EPROM Programmable Controller
Am29CPL142	CMOS 128 x 16 EPROM Programmable Controller
Am29CPL144	CMOS 512 x 18 EPROM Programmable Controller
Am29CPL151	CMOS 64 x 16 EPROM Programmable Controllar-Slim DIP
Am29CPL152	CMOS 128 x 16 EPROM Programmable Controller-Slim DIP
Am29CPL154	CMOS 512 x 16 EPROM Programmable Controller-Slim DIP
Am29C323	CMOS 32-Bit Parallel Multiplier
Am29C325	CMOS 32-Bit Floating-Point Processor
Am29C331	CMOS 16-Bit Sequencer
Am29C332	CMOS 32-Bit Anthmatic Logic Unit
Am29C334	CMOS Four-Port, Dual-Accass Register Fila

CONNECTION DIAGRAM
169-Lead PGA*
Bottom View



09418C-5A
CD011601

*Pinout observed trom pin side ot package.
**Alignment pin (not connected internally).

PGA PIN DESIGNATIONS
(Sorted by Pin No.)

PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
A-1	S_{31}	C-9	V_{CCO}	J-15	$TSEL_1$	R-10	V_{CC}
A-2	F_4	C-10	F_{20}	J-16	$TSEL_0$	R-11	CLOCK
A-3	F_6	C-11	V_{CCO}	J-17	$TSEL_2$	R-12	$\overline{ENF}$
A-4	F_8	C-12	GNDO	K-1	S_9	R-13	$\overline{ENI}$
A-5	F_{10}	C-13	F_{29}	K-2	S_{10}	R-14	FSEL
A-6	F_{12}	C-14	GNDO	K-3	GND	R-15	$RFSEL_1$
A-7	F_{14}	C-15	V_{CCO}	K-15	$QSEL_1$	R-16	$PSEL_3$
A-8	F_{16}	C-16	I_2	K-16	$QSEL_0$	R-17	$PSEL_0$
A-9	F_{18}	C-17	I_6	K-17	$TSEL_3$	T-1	R_{26}
A-10	F_{21}	D-1	S_{24}	L-1	S_6	T-2	R_{23}
A-11	F_{22}	D-2	S_{25}	L-2	S_7	T-3	R_{21}
A-12	F_{24}	D-3	S_{29}	L-3	S_8	T-4	R_{18}
A-13	F_{27}	D-15	I_0	L-15	GNDO	T-5	R_{16}
A-14	F_{28}	D-16	I_3	L-16	$QSEL_3$	T-6	R_{13}
A-15	F_{31}	D-17	I_6	L-17	$QSEL_2$	T-7	R_{10}
A-16	$\overline{SLAVE}$	E-1	S_{21}	M-1	S_5	T-8	R_7
A-17	I_1	E-2	S_{23}	M-2	S_4	T-9	R_5
B-1	S_{30}	E-3	S_{26}	M-3	S_2	T-10	R_3
B-2	F_1	E-15	I_4	M-15	V_{CCO}	T-11	R_0
B-3	F_3	E-16	I_7	M-16	$FLAG_2$	T-12	RM_1
B-4	F_5	E-17	I_9	M-17	$FLAG_1$	T-13	$\overline{ENS}$
B-5	F_7	F-1	S_{18}	N-1	S_3	T-14	$\overline{OES}$
B-6	F_9	F-2	S_{20}	N-2	S_1	T-15	$\overline{ENRF}$
B-7	F_{13}	F-3	S_{22}	N-3	R_{30}	T-16	$RFSEL_0$
B-8	F_{15}	F-15	V_{CC}	N-15	$FLAG_6$	T-17	$PSEL_1$
B-9	F_{17}	F-16	I_{10}	N-16	$FLAG_4$	U-1	R_{25}
B-10	F_{19}	F-17	I_{12}	N-17	$FLAG_3$	U-2	R_{22}
B-11	F_{23}	G-1	S_{15}	P-1	S_0	U-3	R_{19}
B-12	F_{25}	G-2	S_{17}	P-2	R_{29}	U-4	R_{17}
B-13	F_{26}	G-3	S_{19}	P-3	R_{26}	U-5	R_{15}
B-14	F_{30}	G-15	GND	P-15	$PSEL_2$	U-6	R_{14}
B-15	V_{CC}	G-16	I_{11}	P-16	SIGN	U-7	R_{11}
B-16	MSERR	G-17	S/$\overline{DF}$	P-17	$FLAG_5$	U-8	R_9
B-17	I_5	H-1	S_{13}	R-1	R_{31}	U-9	R_8
C-1	S_{27}	H-2	S_{14}	R-2	R_{27}	U-10	R_4
C-2	S_{28}	H-3	S_{16}	R-3	R_{24}	U-11	R_2
C-3	F_0	H-15	S/$\overline{DS}$	R-4	R_{20}	U-12	R_1
C-4	F_2	H-16	I_{13}	R-5	V_{CC}	U-13	RM_0
C-5	V_{CCO}	H-17	S/$\overline{DR}$	R-6	GND	U-14	RM_2
C-6	GNDO	J-1	S_{11}	R-7	R_{12}	U-15	$\overline{ENR}$
C-7	F_{11}	J-2	S_{12}	R-8	R_8	U-16	$\overline{OEF}$
C-8	GNDO	J-3	V_{CC}	R-9	GND	U-17	$RFSEL_2$

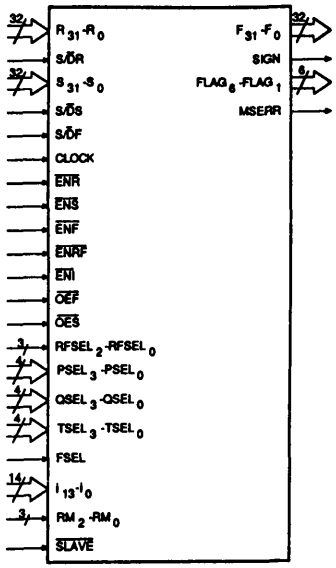
Note: Pin number D-4 = Alignment Pin
V_{CCO} and GNDO are power and ground pins for the output buffers.
V_{CC} and GND are power and ground pins for the rest of the logic.

PGA PIN DESIGNATIONS (Cont'd.)
(Sorted by Pin Name)

PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
R-11	CLOCK	P-17	$FLAG_5$	T-9	R_5	K-1	S_9
R-12	$\overline{ENF}$	N-15	$FLAG_6$	U-9	R_6	K-2	S_{10}
R-13	$\overline{ENI}$	R-14	FSEL	T-8	R_7	J-1	S_{11}
U-15	$\overline{ENR}$	G-15	GND	R-6	R_8	J-2	S_{12}
T-15	$\overline{ENRF}$	K-3	GND	U-8	R_8	H-1	S_{13}
T-13	$\overline{ENS}$	R-6	GND	T-7	R_{10}	H-2	S_{14}
C-3	F_0	R-9	GND	U-7	R_{11}	G-1	S_{15}
B-2	F_1	C-6	GNDO	R-7	R_{12}	H-3	S_{16}
C-4	F_2	C-8	GNDO	T-6	R_{13}	G-2	S_{17}
B-3	F_3	C-12	GNDO	U-6	R_{14}	F-1	S_{18}
A-2	F_4	C-14	GNDO	U-5	R_{15}	G-3	S_{19}
B-4	F_5	L-15	GNDO	T-5	R_{16}	F-2	S_{20}
A-3	F_6	D-15	I0	U-4	R_{17}	E-1	S_{21}
B-5	F_7	A-17	I1	T-4	R_{18}	F-3	S_{22}
A-4	F_8	C-16	I2	U-3	R_{19}	E-2	S_{23}
B-6	F_9	D-16	I3	R-4	R_{20}	D-1	S_{24}
A-5	F_{10}	E-15	I4	T-3	R_{21}	D-2	S_{25}
C-7	F_{11}	B-17	I_5	U-2	R_{22}	E-3	S_{26}
A-6	F_{12}	C-17	I_6	T-2	R_{23}	C-1	S_{27}
B-7	F_{13}	E-16	I_7	R-3	R_{24}	C-2	S_{28}
A-7	F_{14}	D-17	I_8	U-1	R_{25}	D-3	S_{29}
B-8	F_{15}	E-17	I_9	P-3	R_{26}	B-1	S_{30}
A-8	F_{16}	F-16	I_{10}	R-2	R_{27}	A-1	S_{31}
B-9	F_{17}	G-16	I_{11}	T-1	R_{26}	G-17	S/$\overline{DF}$
A-9	F_{18}	F-17	I_{12}	P-2	R_{29}	H-17	S/$\overline{DR}$
B-10	F_{19}	H-16	I_{13}	N-3	R_{30}	H-15	S/$\overline{DS}$
C-10	F_{20}	B-16	MSERR	R-1	R_{31}	P-16	SIGN
A-10	F_{21}	U-16	$\overline{OEF}$	T-16	$RFSEL_0$	A-16	$\overline{SLAVE}$
A-11	F_{22}	T-14	$\overline{OES}$	R-15	$RFSEL_1$	J-16	$TSEL_0$
B-11	F_{23}	R-17	$PSEL_0$	U-17	$RFSEL_2$	J-15	$TSEL_1$
A-12	F_{24}	T-17	$PSEL_1$	U-13	RM_0	J-17	$TSEL_2$
B-12	F_{25}	P-15	$PSEL_2$	T-12	RM_1	K-17	$TSEL_3$
B-13	F_{26}	R-16	$PSEL_3$	U-14	RM_2	B-15	VCC
A-13	F_{27}	K-16	$QSEL_0$	P-1	S_0	F-15	VCC
A-14	F_{28}	K-15	$QSEL_1$	N-2	S_1	J-3	VCC
C-13	F_{29}	L-17	$QSEL_2$	M-3	S_2	R-5	VCC
B-14	F_{30}	L-16	$QSEL_3$	N-1	S_3	R-10	VCC
A-15	F_{31}	T-11	R_0	M-2	S_4	C-5	VCCO
M-17	$FLAG_1$	U-12	R_1	M-1	S_5	C-9	VCCO
M-16	$FLAG_2$	U-11	R_2	L-3	S_6	C-11	VCCO
N-17	$FLAG_3$	T-10	R_3	L-2	S_7	C-15	VCCO
N-16	$FLAG_4$	U-10	R_4	L-1	S_8	M-15	VCCO

Note: Pin number D-4 = Alignment Pin
V_{CCO} and GNDO are power and ground pins for the output buffers.
V_{CC} and GND are power and ground pins for the rest of the logic.

LOGIC SYMBOL



LS003261

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:

a. Device Number
b. Speed Option (if applicable)
c. Package Type
d. Temperature Range
e. Optional Processing

AM29C327 -1 G C

e. **OPTIONAL PROCESSING**
Blank = Standard processing

d. **TEMPERATURE RANGE**
C = Commercial (0 to +70°C)

c. **PACKAGE TYPE**
G = 169-Lead Pin Grid Array (CGX169)

b. **SPEED OPTION**
Blank = Base Speed
-1 = Speed Select

a. **DEVICE NUMBER/DESCRIPTION**
Am29C327
Double-Precision Floating-Point Processor

Valid Combinations	
AM29C327	GC
AM29C327-1	

Valid Combinations

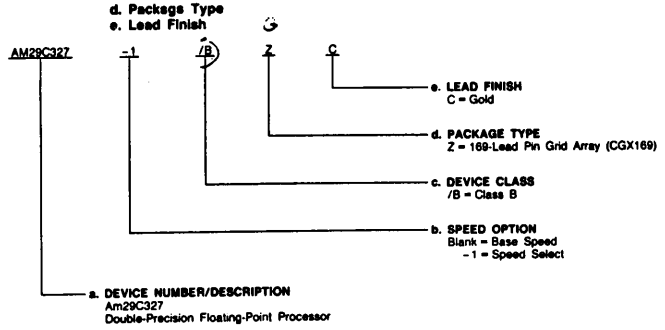
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

MILITARY ORDERING INFORMATION

APL Products

AMD standard products for Aerospace and Defence applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) is formed by a combination of:

a. **Device Number**
b. **Speed Option** (if applicable)
c. **Device Class**
d. **Package Type**
e. **Lead Finish**



Valid Combinations	
AM29C327	/BZC
AM29C327-1	

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

Group A Tests

Group A tests consist of Subgroups 1, 2, 3, 7, 8, 9, 10, 11.

PIN DESCRIPTION

CLOCK Clock (Input)
Clock input to all registers. The Am29C327 is fully static — no data is lost from the internal registers if the clock is stopped for an extended period.

$\overline{ENF}$ F-Register Enable (Input; Active LOW)
When $\overline{ENF}$ is HIGH, the contents of the F-register are static. When $\overline{ENF}$ is LOW, the 64-bit ALU output is clocked into the F-register on the next LOW-to-HIGH transition of the CLOCK input. As described in the Mode Register Description section, the F-register can be made transperent by setting the mode register bit M17 appropriately, in which case $\overline{ENF}$ has no effect. This input is not clocked into the instruction register, and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the F-register.

$\overline{ENI}$ Instruction Register Enable (Input; Active LOW)
When $\overline{ENI}$ is HIGH, the contants of the instruction register are static. When $\overline{ENI}$ is LOW, the 30-bit instruction word, comprising the fields $PSEL_{3-0}$, $QSEL_{3-0}$, $TSEL_{3-0}$, RM_{2-0}, $S/\overline{DF}$ and I_{13-0}, is clocked into the instruction register on the next LOW-to-HIGH transition of the CLOCK input. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the instruction register.

$\overline{ENR}$ R-Register Enable (Input; Active LOW)
When $\overline{ENR}$ is HIGH, the contents of the R-register are static. When $\overline{ENR}$ is LOW, a new 64-bit operand, together with the precision control input $S/\overline{DR}$, is clocked into the 65-bit R-register on the next LOW-to-HIGH transition of the CLOCK input. As described in the Input Modes section, the user can select from eight different input modes, as appropriate for the system environment. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the R-register.

$\overline{ENRF}$ Register File Enable (Input; Active LOW)
When $\overline{ENRF}$ is HIGH, the contents of the register file are static. When $\overline{ENRF}$ is LOW, the 64-bit ALU result, together with a "tag" indicating its precision, is clocked into one of the 65-bit registers RF7 to RF0 on the next LOW-to-HIGH transition of the CLOCK input. The inputs $RFSEL_{2-0}$ determine which of the eight registers in the register file is the destination for the ALU result and its precision tag. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the register file.

$\overline{ENS}$ S-Register Enable (Input; Active LOW)
When $\overline{ENS}$ is HIGH, the contents of the S-register are static. When $\overline{ENS}$ is LOW, a new 64-bit operand, together with the precision control input $S/\overline{DS}$, is clocked into the 65-bit S-register on the next LOW-to-HIGH transition of the CLOCK input. As described in the Input Modes section, the user can select from eight different input modes, as appropriate for the system environment. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the S-register.

F_{31-0} Output Bus (Bidirectional)
The 32-bit output bus is bidirectional to support Master/Slave checking.

$FLAG_{6-1}$ Flag Outputs (Bidirectional)
The six flag outputs $FLAG_6$-$FLAG_1$ report the status of the previous ALU operation. The outputs are bidirectional to support Master/Slave checking.

FSEL Output Multiplexer Control (Input)
When FSEL is HIGH, the most-significant 32 bits of the 64-bit F-register are connected to the drivers on the F_{31-0} output bus. When FSEL is LOW, the least-significant 32 bits of the 64-bit F-register are connected to the drivers on the F_{31-0} output bus. The state of this input pin may be changed at twice the rate of the CLOCK input, to allow a full 64-bit result to be output from the Am29C327 in a single clock cycle. This input is not clocked into the instruction register and must be valid while the required data is accessed via the output bus.

I_{13-0} ALU Instruction (Input)
I_{13-0} determine the ALU instruction to be executed in the next cycle. The inputs are clocked into the instruction register under the control of the $\overline{ENI}$ input.

MSERR Master/Slave Error (Output)
The MSERR output is asserted (HIGH) whenever a Master/Slave error is detected on any enabled output, i.e. F_{31-0} if $\overline{OEF}$ is LOW, SIGN and $FLAG_{6-1}$ if $\overline{OES}$ is LOW.

$\overline{OEF}$ F-Output Enable (Input; Active LOW)
When $\overline{OEF}$ is HIGH, the F_{31-0} output bus assumes a high-impedance state. When $\overline{OEF}$ is LOW (and $\overline{SLAVE}$ is HIGH, specifying "Master" mode), the F_{31-0} output drivers are enabled. This input is not clocked into the instruction register and must be valid at all times.

$\overline{OES}$ Flag Output Enable (Input; Active LOW)
When $\overline{OES}$ is HIGH, the 7 outputs $FLAG_{6-1}$ and SIGN assume a high-impedance state. When $\overline{OES}$ is LOW (and $\overline{SLAVE}$ is HIGH, specifying "Master" mode), the output drivers for the $FLAG_{6-1}$ and SIGN outputs are enabled. This input is not clocked into the instruction register and must be valid at all times.

$PSEL_{3-0}$ P-Input Multiplexer Control (Input)
The $PSEL_{3-0}$ inputs control the P-Input Multiplexer, selecting the source of operands for the P-Input of the ALU. The $PSEL_{3-0}$ inputs are clocked into the instruction register under the control of the $\overline{ENI}$ input.

$QSEL_{3-0}$ Q-Input Multiplexer Control (Input)
The $QSEL_{3-0}$ inputs control the Q-Input Multiplexer, selecting the source of operands for the Q-Input of the ALU. The $QSEL_{3-0}$ inputs are clocked into the instruction register under the control of the $\overline{ENI}$ input.

R_{31-0} R-Input Bus (Input)
The 32-bit R-Input bus, R_{31-0}, is used to load operands into one or both of the input registers, R and S. It is also used to load data into the 32-bit mode register.

$RFSEL_{2-0}$ Register File Destination Select (Input)
The $RFSEL_{2-0}$ inputs select which of the registers RF7 through RF0 is the destination for the ALU result. Operands are clocked into the selected register only when the $\overline{ENRF}$ input is LOW. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the register file.

RM_{2-0} Rounding Mode Select (Input)
The RM_{2-0} inputs select which of the six available rounding modes is to be applied to the next ALU operation. Rounding is discussed in Appendix B. The RM_{2-0} inputs are clocked into the instruction register under the control of the $\overline{ENI}$ input.

S_{31-0} S-Input Bus (Input)
The 32-bit S-Input bus, S_{31-0}, is used to load operands into one or both of the input registers, R and S.

S/$\overline{D}$F F-Precision Control (Input)
When S/$\overline{D}$F is HIGH, the next ALU operation produces a single-precision (32-bit) result. When S/$\overline{D}$F is LOW, the next ALU operation produces a double-precision (64-bit) result. The S/$\overline{D}$F input is clocked into the instruction register under the control of the $\overline{ENI}$ input.

S/$\overline{D}$R R-Precision Control (Input)
When S/$\overline{D}$R is HIGH, the data clocked into the R-register is treated as single-precision (32-bit) by the processor. When S/$\overline{D}$R is LOW, the data clocked into the R-register is treated es double-precision (64-bit) by the processor. The S/$\overline{D}$R input is clocked into the 85th bit of the R-register as the "precision tag" for the R-operand, under the control of the $\overline{ENR}$ input.

S/$\overline{D}$S S-Precision Control (Input)
When S/$\overline{D}$S is HIGH, the date clocked Into the S-register ie treated as single-precision (32-bit) by the processor. When S/$\overline{D}$S is LOW, the data clocked Into the S-register is treated as double-precision (64-bit) by the processor. The S/$\overline{D}$S input is clocked into the 65th bit of the S-register as the "precision tag" for the S-operand, under the control of the $\overline{ENS}$ input.

SIGN Sign Flag (Bidirectional)
If the result of the previous ALU operation was negative, the SIGN output is HIGH (provided that $\overline{OES}$ is LOW). If the result of the previous ALU operation was not negative, the SIGN output is LOW (provided that $\overline{OES}$ is LOW). The output Is bidirectional to support Master/Slave checking.

$\overline{SLAVE}$ Master/Slave Mode Select (Input; Active LOW)
When $\overline{SLAVE}$ is HIGH, the "Master" mode of operation is selected. When $\overline{SLAVE}$ is LOW, the "Slave" mode of operation ie selected and all outputs except MSERR are disabled (high-impedance). This input is not clocked into the instruction register and must be valid at all times.

$TSEL_{3-0}$ T-Input Multiplexer Control (Input)
The $TSEL_{3-0}$ inputs control the T-Input Multiplexer, selecting the source of operands for the T-Input of the ALU. The $TSEL_{3-0}$ inputs are clocked into the instruction register under the control of the $\overline{ENI}$ input.

V_{CC} GND Power
Power supply pins for the internal logic.

V_{CCO} GNDO Power
Power supply pins for the output buffers.

FUNCTIONAL DESCRIPTION

Overview

The Am29C327 is a high-performance, single-chip, double-precision floating-point ˙ ˙ocessor.

Architecture

The Am29C327 comprises a high-speed ALU, a 64-bit data path, and control circuitry.

The core of the Am29C327 is a 64-bit floating-point/integer ALU. This ALU takes operands from three 64-bit Input ports and performs the selected operation, placing the resulf on a 64-bit output port. Thirteen ALU flaga report operation status via the $FLAG_{6-1}$ and SIGN outputs. The ALU is completely combinatorial for minimum latency; optional pipelining ie available to boost throughput for array operations.

The data path consists of the 32-bit input buses R and S; two 64-bit input operand registers; an 8-by-64-bit register file for storege of intermediate results; three operend-selection multiplexers thet provide for orthogonal selection of input operands; a 64-bit output register; and an output multiplexer that permits the selection of 32 MSBs, or 32 LSBs of data. Input operands enter the processor through the R and S buses, and are then demultiplexed and buffered for subsequent storage In registers R and S. The operand selection multiplexers route the operands to the ALU. Operation results are stored in register F, and leave the device on the 32-bit output bus F. The results can also be atored in the register file for use in subsequent operations.

Instruction Set

The Am29C327 implements 58 arithmetic and logical instructions. Thirty-five instructions operate on floating-point numbers; these instructions fall into the following categories:

- Addition/subtraction
- Multiplication
- Multiplication-accumulation
- Comparison
- Selecting the maximum or minimum of two numbers
- Rounding to integral value
- Absolute value, negation, pass
- Reciprocal seed generation
- Conversion between any of the supported floating-point formats
- Conversion of e floating-point number to an Integer format, with or without a scale factor

By concatenating these operationa, the user can also perform division, square-root extraction, polynomial evaluation, and other functions not implemented directly.

Twenty-two instructions operate on integers, and belong to the following general categories:

- Addition/subtraction
- Multiplication
- Comparison
- Selecting the maximum or minimum of two numbers
- Absolute value, negation, pass
- Logical operations; e.g., AND, OR, XOR, NOT
- Arithmetic, logical, and funnel shifts
- Conversion between single- and double-precision integer formats
- Conversion of an integer number to a floating-point format, with or without a scale factor

One special instruction is provided to move data from the P-Port to the F-Port, and another to load the mode register.

Mixed-Precision Operations

All Am29C327 instructions, floating-point or integer, can be performed with either single- or double-precision operands. In addition, the user can elect to mix precisions within an operation. All operations are performed in double-precision internally; the user specifies the precisions of the input operands and the required precision for the output operend. The necessary precision conversions are made in concert with the selected operation, with no additionel cycle-time overhead.

I/O Modes

The Am29C327 supports eight I/O modes that afford flexible interface to e variety of 32- end 64-bit systems.

Fault Detection Features

The Am29C327 contains special comparison hardware to ellow the operetion of two processors in parallel, with one processor (the slave) checking the results produced by the other (the master). This feature is of particular importance in the design of high-reliability systems.

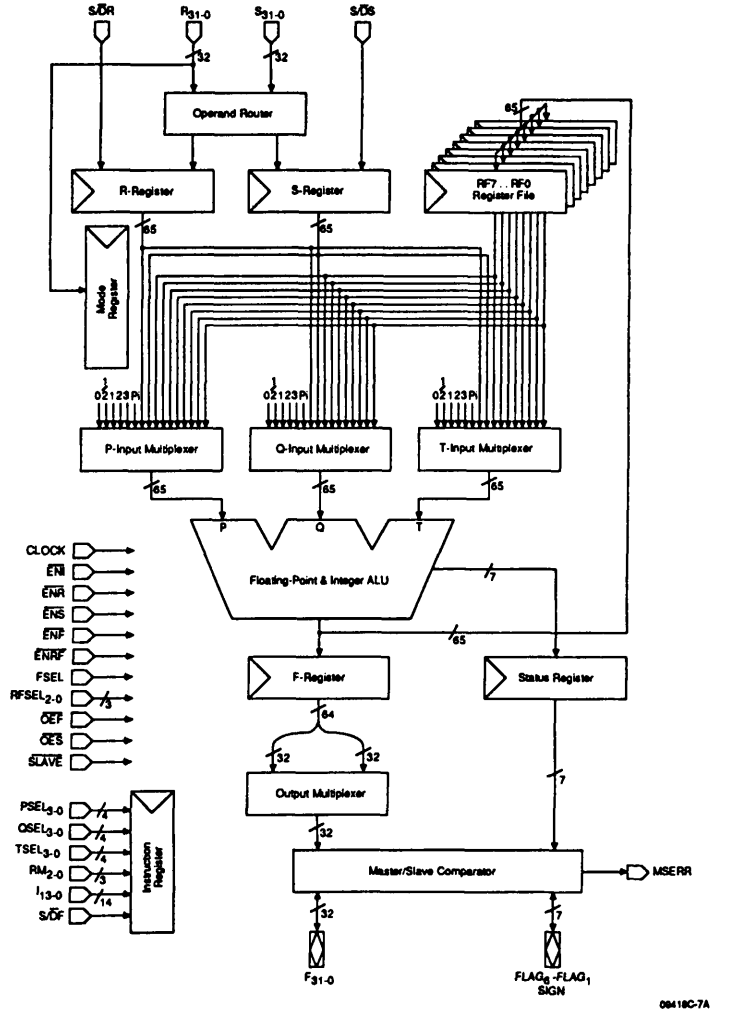


Figure 1. Block Diagram – Am29C327 Double-Precision Floating-Point Processor

Block Diagram Description

A block diagram of the Am29C327 is shown in Figure 1. The Am29C327 comprises the input registers, the operand selection multiplexers, the instruction register, the ALU, the output register/the register file, the status register, the output selection multiplexer, the mode register, and the master/slave comparator.

Input Registers/Input Modes

Operands are loaded into the processor through the R and S buses, and are then demultiplexed and buffered for subsequent storage in the 65-bit registers R and S. Input operands may be either single-precision (32-bit) or double-precision (64-bit) as specified by S/D̄R and S/D̄S. Accompanying the input registers are two 32-bit temporary registers, that allow for the overlapping of operand transfers and ALU operations. This arrangement of temporary registers and demultiplexers permits data and corresponding precision bit S/D̄R or S/D̄S to be loaded into the 65-bit R register and 65-bit S register via one of the eight input modes:

1. 32-bit-bus, double-cycle, LSWs first
2. 32-bit-bus, double-cycle, MSWs first
3. 32-bit-bus, single-cycle, LSWs first
4. 32-bit-bus, single-cycle, MSWs first
5. 64-bit-bus, double-cycle, R first
6. 64-bit-bus, double-cycle, S first
7. 64-bit-bus, single-cycle, R first
8. 64-bit-bus, single-cycle, S first

These modes are described in detail in the Input Modes section.

Operand Selection Multiplexers

The operand selection multiplexers route operands to the ALU. These multiplexers, as well as selecting operands from input registers R and S and register file locations RF7 – RF0, also have access to a set of constants (0, 0.5, 1, 2, 3, Pi). These constants are double-precision preprogrammed numbers for use in ALU operations, and are automatically provided in the appropriate floating-point or integer format.

Instruction Register

The instruction register stores a 32-bit word specifying the current processor operation. Included in the instruction word are fields that specify the P, Q, and T multiplexer selects, the rounding modes; the core operation to be performed by the ALU; sign-change controls for ALU input and result operands; and the single/double-precision control for the output operand. The multiplexer select controls and the instruction word are described in detail in the Instruction Set section; Rounding modes are described in Appendix B.

ALU

The ALU is a combinatorial arithmetic/logic unit that performs a large repertoire of floating-point and integer operations. The ALU has three operand inputs. Some operations require a single input operand, for example, conversion operations. Others, such as addition and multiplication, require two input operands. The multiplication-accumulation and funnel shift operations require three input operands. Many ALU arithmetic operations allow for the independent control of operand signs, thus greatly increasing the number of arithmetic expressions that can be evaluated in a single ALU pass.

The ALU can be configured in either a flow-through mode, for which the ALU is completely combinatorial, or a pipelined mode, for which ALU operations incur one or two pipeline delays, but which results in a higher throughput than flow-through mode.

A detailed description of ALU operations appears in the Instruction Set section.

Output Register/Register File

The results of the operations performed by the ALU are stored in the 64-bit output register F. Results can also be stored in the 8-by-64-bit register file for use in subsequent operations. Each register file location contains a 65th bit indicating the precision of the operand stored in that location, thus permitting the ALU to correctly process the operand in subsequent operations.

Status Register

The status register is a 7-bit register that stores flags pertaining to the most recently performed operation. A detailed description is provided in the Instruction Set section.

Output Multiplexer

The output multiplexer routes operation results to the F bus. This multiplexer selects either the 32 MSBs or the 32 LSBs of the data stored in the output register.

Master/Slave Comparator

Each Am29C327 output signal has associated logic that compares that signal with the signal that the processor is providing internally to the output driver; any discrepancies are indicated by assertion of signal MSERR.

For a single processor, this output comparison detects short circuits in output signals or defective output drivers, but does not detect open circuits. It is possible to connect a second processor in parallel with the first, with the second processor's outputs disabled by assertion of signal SLAVE. The second processor detects open-circuit signals, as well as providing a check of the outputs of the first.

Mode Register

The mode register contains processor control parameters that are changed infrequently. The 32-bit mode word is loaded into the register via the R bus. A detailed description of the mode register is provided in the Mode Register Description section.

Mode Register Description

The 32-bit mode register contains parameters specifying the overall opereting mode of the Am29C327. These parameters are typically not changed on an instruction-by-instruction basis. The register is loaded from the R-port R_{31-0}, using the "Load Mode Register" instruction. This section provides a comprehensive explanetion of the function of each field within the register.

Bits M31-M21

Reserved for factory test and future upgrades. Must be set to logic 0.

Bits M20-M19 — Pipeline Mode Select

This field determines whether the ALU operates in flow-through mode or pipelined mode, and specifies whether pipelined multiply-eccumulate operations are single-pipelined or double-pipelined:

M20	M19	
0	X	The ALU operates in flow-through mode for all operations — all pipeline registers are transparent.
1	0	The ALU operates in single-pipelined mode for all operations.
t	t	The ALU operates in double-pipelined mode for multiply-accumulate operations. The ALU operates in single-pipelined mode for all other operations.

A detailed description of . .o pipeline modes is contained in the Pipelining of Operations section.

Bit M18 — Statue Register Feedthrough Enable

If M16 is HIGH, the 7-bit stetus register is made transparent and operetes in flow-through mode. This mode is generally used when it is necessary to minimize the overall latency of the processor. If M16 is LOW, the status register operates in clocked mode, stetus information being clocked into the register from the ALU on every rising edge of the clock input. This mode is generally used when it is necessary to maximize the overall throughput of the processor.

Bit M17 - F-Register Feedthrough Enable

If M17 is HIGH, the 64-bit F-register is made trensparent end operetes in flow-through mode. This mode is generally used when it is necessary to minimize the overall latency of the processor. If M17 is LOW, the F-register operates in clocked mode, ALU results being clocked into the register from the ALU on every rising edge of the clock input, providnd that $\overline{ENF}$ is LOW. This mode is generally used when it is necessary to maximize the overell throughput of the processor.

Bits M16-M14 — Input Mode Select

This field determines which of the eight availeble modes is used to input operands to the R-register and S-register:

M16	M15	M14	Input Mode
0	0	0	32-bit bus, single-cycle, LSW first.
0	0	t	32-bit bus, single-cycle, MSW first.
0	t	0	32-bit bus, double-cycle, LSW first.
0	t	t	32-bit bus, double-cycle, MSW first.
t	0	0	64-bit bus, single-cycle, R first.
t	0	t	64-bit bus, single-cycle, S first.
t	t	0	64-bit bus, double-cycle, R first.
t	t	1	64-bit bus, double-cycle, S first.

A detailod description of the input modes is contained in the Input Mode section.

Bits M13-M12 — Integer Multiplication Format Adjuet

This field determines the output format for integer multiplication operetions, selecting either the MSBs or the LSBs of the product and specifying whether or not format-adjusting (i.e., shifting the product one place left before the MSBs/LSBs selection) is performed:

M13	M12	Integer Multiplication Output
0	0	LSBs selected, no format-adjust performed.
0	t	LSBs selected, format-adjust performed.
1	0	MSBs selected, no format-adjust performed.
t	1	MSBs selected, format-adjust performed.

This field has no effect on operetions other than integer multiplications.

Bit M11 — Integer Multiplication Signed/Unaigned

If Mtt is HIGH, the input operands for integer multiplicetions are treated as two's-complement (signed) operends end two's-complement multiplications are performed. If Mtt is LOW, the input operands for integer multiplicetions are treated as unsigned operends and unsigned multiplications are performed.

This bit has no effect on operations other than integer multiplications.

Bit M10

Reserved for factory test and future upgrades. Must be set to logic 0.

Bit M9 — IBM Underflow Mask Enable

If M9 is HIGH, underflowed results in IBM format are output with the (underflowed) exponent increased by t26 to bring its value into the representable range. If M9 is LOW, underflowed results in IBM format are repleced by an IBM "True Zero" (sign = 0, exponent = 0, fraction = 0).

This bit hes no effect on operetions for which the result format is not IBM.

Bit M8 — IBM Significance Mask Enable

If M8 is HIGH, results in IBM format that contain a non-zero exponent and e zero fraction (known es IBM "Floating-Point Zeros") are output unchanged. If M6 is LOW, results in IBM format that contain a non-zero exponent and e zero frection are replaced by an IBM "True Zero" (sign = 0, exponent = 0, fraction = 0).

This bit has no effect on operetions for which the result format is not IBM.

Bit M7 — IEEE Sudden Underflow Enable

If M7 is HIGH (and M6 is LOW, disabling IEEE traps), results in IEEE formet that are denormalized are replaced by e zero of the same sign ("Sudden Underflow" mode). If M7 is LOW (and M6 is LOW, disabling IEEE treps), results in IEEE format that are denormalized ere output unchanged as valid denormelized numbers ("Gradual Underflow" mode).

This bit hes no effect on operetions for which the result format is not IEEE or on operetions for which the result format is IEEE but treps are enebled.

Bit M6 — IEEE Trepped Operetion Enable

If M6 is HIGH, the exponents of IEEE overflowed results are reduced by t92 (single-precision) or t536 (double-precision) to bring them into the representable range, the signs end frections being unchanged, regardless of the setting of M4 (Saturete Enable). Similerly, if M6 is HIGH, the exponents of IEEE underflowed and denormalized results are increased by t92 (single-precision) or t536 (double-precision) to bring them into the representable range, the signs and fractions being unchanged, regerdless of the setting of M7 (Sudden Underflow Enable). If M6 is LOW, the final results of overflowed and underflowed IEEE operations are determined by bits M4 (Seturate Enable) and M7 (Sudden Underflow Enable) respectively.

This bit hes no effect on operations for which the result format is not IEEE.

Bit M5 — IEEE Affine/Projective Mode Select

If M5 is HIGH, IEEE infinities are interpreted in the "Affine" sense for addition, comperison end multiply-accumulate operations. If M5 is LOW, IEEE infinities are interpreted in the "Projective" sense for addition, comparison and multiply-eccumulate operations. The difference between affine and projective interpretetions is summarized below.

Affine and Projective Interpretations of Infinities

Operation	Affine Result & Flags	Projective Result & Flags
(+ oo) + (+ oo)	+ oo; No flegs	Quiet Nen; Flegs I, R
(- oo) + (- oo)	- oo; No flags	Quiet Nen; Flags I, R

Flag I is the "Invalid Operetion" flag. Flag R is the "Reserved Operand" flag.

This bit has no effect on bese operations other then IEEE addition, IEEE comparison and IEEE multiply-accumulate.

Bit M4 - Saturate Enable

If M4 is HIGH, overflowed results ere replaced by the largest representeble number in the result formet, with the same sign as the overflowed result, unless the result format is IEEE and M6 is HIGH, enebling IEEE trapped operation. If M4 is LOW, overflowed results ere repleced by infinities (IEEE results, provided that M6 is LOW, disebling trapped operations), repleced by DEC Reserved Operands (DEC results) or left unchenged (IBM end integer results).

When IEEE traps are enabled (M6 is HIGH), this bit has no effect on operetions for which the result formet is IEEE.

Bits M3-M2 — Alternate Floeting-Point Formet Select

This field determines the elternate floeting-point format:

M3	M2	Alternate Floeting-Point Formet (Double/Single)
0	0	IEEE (IEEE double-precision/IEEE single-precision)
0	t	DEC D (DEC D/DEC F)
1	0	DEC G (DEC G/DEC F)
t	1	IBM (tBM double-precision/tBM single-precision)

The alternate floating-point formet is used for floeting-point conversion operations to specify the destination format for the operation "Convert T to Alternete Format" and to specify the source format for the operetion "Convert T from Alternete Formet".

The floeting-point formats ere specified eccording to their double-precision names, since the informetion regerding the precisions of operands is provided to the processor via the S/$\overline{D}$R, S/$\overline{D}$S and S/$\overline{D}$F inputs. Note that the two, distinct, double-precison DEC formats, DEC D and DEC G, have the same single-precision format, DEC F.

Bits M1-M0 — Primary Floeting-Point Format Select

This field determines the primary floating-point format:

M1	M0	Primary Floating-Point Format (Double/Single)
0	0	IEEE (IEEE double-precision/IEEE single-precision)
0	t	DEC D (DEC D/DEC F)
1	0	DEC G (DEC G/DEC F)
t	1	IBM (IBM double-precision/IBM single-precision)

The primary floating-point formet is the format in which all floating-point inputs ere assumed to be represented end in which ell floating-point results ere generated, taking into account specified precisions, with the following exceptions: the T-input for the operetion "Convert T from Alternate Format" is assumed to be the alternate floeting-point format and the result of the operation "Convert T to Alternete Format" is generated in the alternate floating-point formet, egain taking into account specified precisions.

The floating-point formets are specified according to their double-precision names, since the informetion regerding the precisions of operands is provided to the processor via the S/$\overline{D}$R, S/$\overline{D}$S and S/$\overline{D}$F inputs. Note that the two, distinct, double-precision DEC formets, DEC D and DEC G, heve the same single-precision format, DEC F.

The mode register bits, which affect the arithmetic result of en operation rather than controlling deta movement within the processor, are bits M13 through M4. The teble below indicates the settings thet should be used for these t0 bits to ensure strict conformance with the IEEE, DEC, and IBM standerds. As regards IEEE treps end IBM mesks, the relevent standards specify thet these bits be set es dicteted by the application. In ell cases, of course, bits M1 end M0 must be set to specify the desired primary format.

It should be noted that the IEEE, DEC, and IBM standerds do not explicitly define conversion operetions between floeting-point formats. When executing these operations, the eppropriate settings for the mode register ere, in generel, determined by the system environment and the alogorithms to be executed. Bits M3 end M2 must be set to specify the required alternate floating-point formet.

Mode Settings for Exact Compliance with Standards

Bit	Function	IEEE	DEC	IBM
13-t2	Int FMTADJ	X	X	X
11	Int SIGNSEL	X	X	X
9	IBM UNFMASK	X	X	As Reqd.
8	IBM SIGMASK	X	X	As Reqd.
7	IEEE SUEN	0	X	X
6	IEEE TRAPEN	As Reqd.	X	X
5	IEEE AFF/$\overline{PROJ}$	t	X	X
4	SATEN	0	0	0

"0" = set to logic 0.
"1" = set to logic 1.
"X" = don't cere.

Input Modes

The Am29C327 supports a total of eight input modes for loading data into the R and S registers.

The 32-bit bus modes allow the user to connect each input port ($R_{31} - R_0$ and $S_{31} - S_0$) to a separate 32-bit bus. 64-bit operands can then be loaded by placing the MSBs and LSBs alternately on the appropriate ports. In the 64-bit bus modes, the two input ports are configured internally as a single 64-bit port. The Am29C327 may then be connected directly to a 64-bit bus, and 64-bit operands may be loaded in a single operation. Either the 32-bit bus modes or the 64-bit bus modes may be used regardless of the precision of the operands being transferred — the choice of input modes will in practice be determined by the system into which the Am29C327 is to be integrated.

Single-cycle input modes allow two 64-bit operands to be loaded in a single clock cycle. This necessitates driving the input buses at twice the speed of the Am29C327. In systems where this is not practical, the double-cycle modes allow the loading of one 64-bit operand (or two 32-bit operands) per clock cycle.

Data may be loaded from the input buses to the R register and S register using one of the eight input modes:

1. 32-Bit Bus, Single-Cycle, LSWs First
2. 32-Bit Bus, Single-Cycle, MSWs First
3. 32-Bit Bus, Double-Cycle, LSWs First
4. 32-Bit Bus, Double-Cycle, MSWs First
5. 64-Bit Bus, Single-Cycle, R First
6. 64-Bit Bus, Single-Cycle, S First
7. 64-Bit Bus, Double-Cycle, R First
8. 64-Bit Bus, Double-Cycle, S First

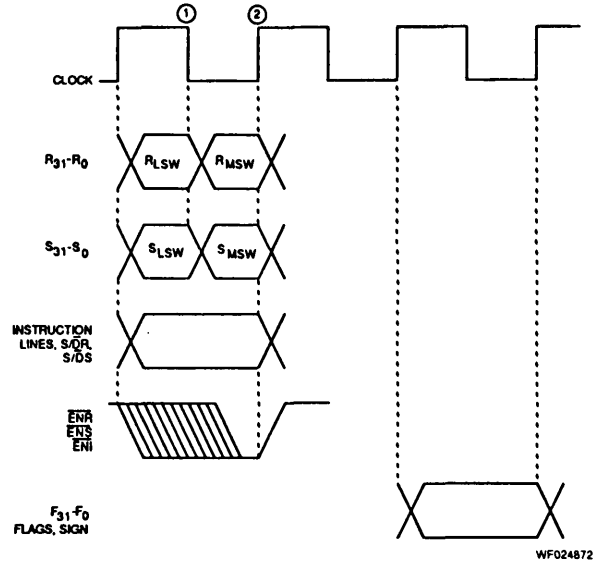
The choice of input mode is determined by mode register bits M16 – M14.

In order to permit the loading of new operands to be overlapped with the execution of a current operation, temporary registers are provided within the "operend router" block (shown in Figure 1). The operation of these temporary registers is transparent to the user. The conditions under which they are loaded depends on the input mode selected.

The eight input modes are described in the following pages.

32-Bit Bus, Single-Cycle, LSW First (M16 = 0, M15 = 0, M14 = 0)

In this mode, the two halves of the 64-bit R operand are placed on the R-port in successive half-cycles, with the S operand similarly placed on the S-port. After one complete cycle, the R and S registers contain the R and S operands, respectively.



Timing of Operations with Input Mode 1 (32-Bit Bus, Single-Cycle, LSW First)*

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every HIGH-to-LOW clock transition.

At 1, the least-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register. Both words are loaded on the HIGH-to-LOW transition of the clock.

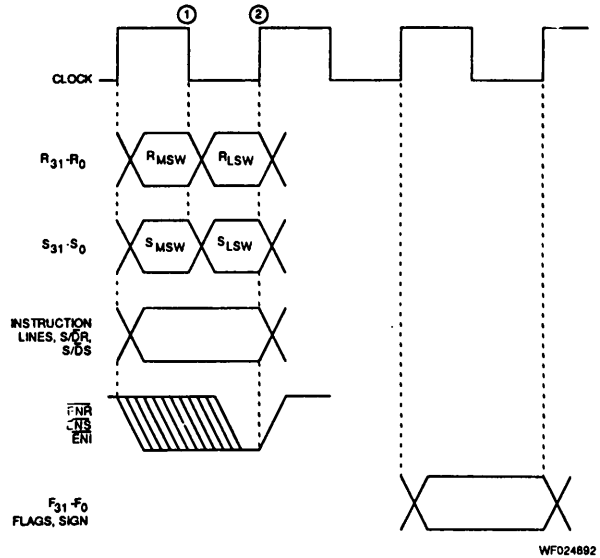
At 2, the most-significant 32 bits of the R operand are loaded from the R-input port into the most-significant half of the R register, and the most-significant 32 bits of the S operand are loaded from the S-input port into the most-significant half of the S register.

At the same time, at 2, the output of the R-temp register is loaded into the least-significant half of the R register, and the output of the S-temp register is loaded into the least-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/$\overline{D}$R = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/$\overline{D}$R = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operend, if it is a single-precision operend.

32-Bit Bus, Single-Cycle, MSW First (M16 = 0, M15 = 0, M14 = 1)

In this mode, the two halves of the 64-bit R operand are placed on the R-port in successive half-cycles, with the S operand similarly placed on the S-port. After one complete cycle, the R and S registers contain the R and S operands, respectively.



Timing of Operations with Input Mode 2 (32-Bit Bus, Single-Cycle, MSW First)*

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every HIGH-to-LOW clock transition.

At 1, the most-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the most-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register. Both words are loaded on the HIGH-to-LOW transition of the clock.

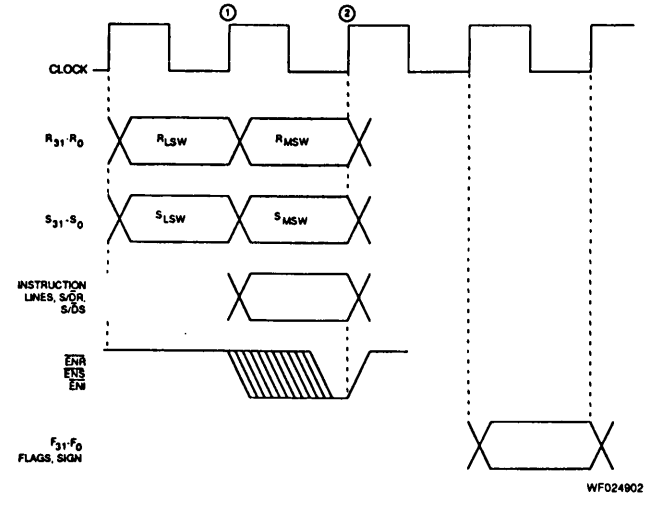
At 2, the least-significant 32 bits of the R operand are loaded from the R-input port into the least-significant half of the R register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the least-significant half of the S register.

At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the R register, and the output of the S-temp register is loaded into the most-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/$\overline{DR}$ = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/$\overline{DR}$ = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

32-Bit Bus, Double-Cycle, LSW First (M16 = 0, M15 = 1, M14 = 0)

In this mode, the two halves of the 64-bit R operand are placed on the R-port in successive cycles, with the S operand similarly placed on the S-port. After two cycles, the R and S registers contain the R and S operands, respectively.



Timing of Operations with Input Mode 3 (32-Bit Bus, Double-Cycle, LSW First)*

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every LOW-to-HIGH clock transition.

At 1, the least-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register.

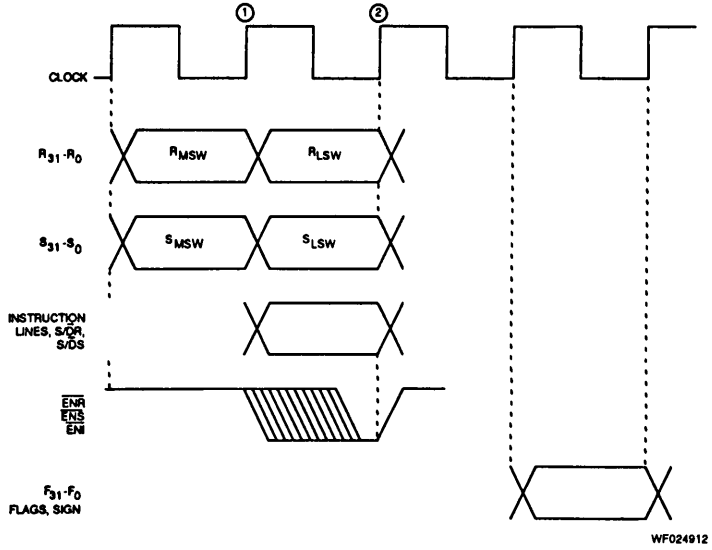
At 2, the most-significant 32 bits of the R operand are loaded from the R-input port into the most-significant half of the R register, and the most-significant 32 bits of the S operand are loaded from the S-input port into the most-significant half of the S register.

At the same time, at 2, the output of the R-temp register is loaded into the least-significant half of the R register, and the output of the S-temp register is loaded into the least-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/$\overline{DR}$ = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/$\overline{DR}$ = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

32-Bit Bus, Double-Cycle, MSW First (M16 = 0, M15 = 1, M14 = 1)

In this mode, the two halves of the 64-bit R operand are placed on the R-port in successive cycles, with the S operend similarly placed on the S-port. After two cycles, the R and S registers contain the R and S operands, respectively.



Timing of Operations with Input Mode 4 (32-Bit Bus, Double-Cycle, MSW First)*

*Assumes processor flow-through mode, F register, end status register clocked.

The temporery registers are clocked on every LOW-to-HIGH clock transition.

At 1, the most-significent 32 bits of the R operend are loeded from the R-input port into tha R-temp register, end the most-significant 32 bits of the S operend are loaded from the S-input port into the S-tamp register.

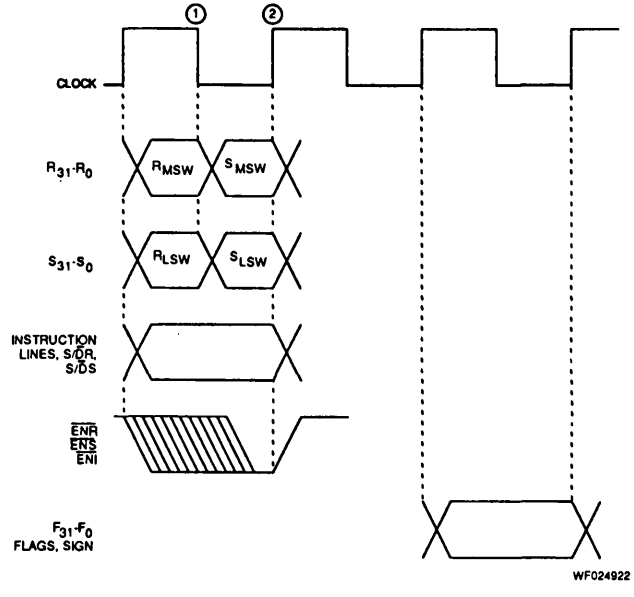
At 2, the least-significant 32 bits of the R operend are loaded from the R-input port fnto the least-significant half of the R register, and the leest-significant 32 bits of the S operand are loaded from the S-input port into the least-significant half of the S register.

At the same time, at 2, the output of the R-temp register is loaded into the most-significent helf of the R register, and the output of the S-temp register is loeded into the most-significant half of the S register.

If the R-operend is e single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/D̄R = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is e singla-pracision (32-bit) Integer operand, it is Input in the R-LSW position (with S/D̄R = HIGH) and the deta in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

64-Bit Bus, Single-Cycle, R First (M16 = 1, M15 = 0, M14 = 0)

In this mode, the MSW of the 64-bit R operand is placed on the R-port and the LSW on the S-port. Both halfwords are loaded in the first half cycle. Similarly, the two halves of the S operand are loaded in the second half cycle. After one full cycle, the R and S registars contain the R and S operands, respectively.



Timing of Operations with Input Mode 5 (64-Bit Bus, Single-Cycle, R First)*

*Assumes processor flow-through mode, F register, end status register clocked.

The temporary ragisters ara clocked on every HIGH-to-LOW clock transition.

At 1, tha most-significant 32 bits of the R operand are loaded from the R-input port into the R-tamp register, and the leest-significant 32 bits of the R operand are loeded from the S-input port into tha S-temp register.

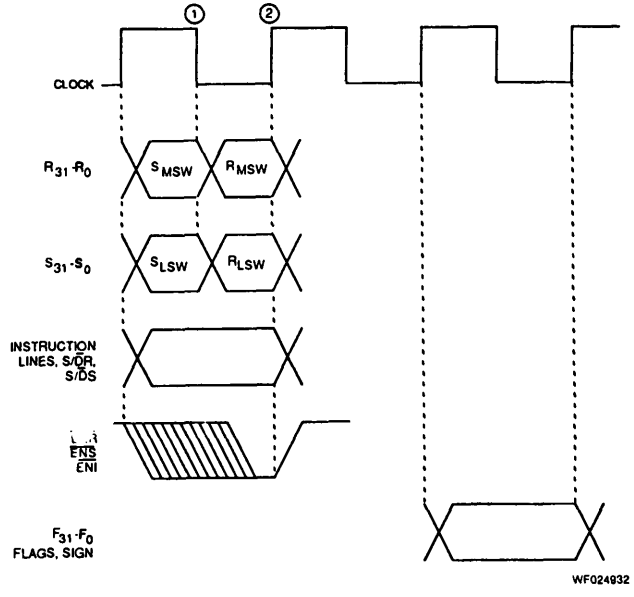
At 2, the most-significent 32 bits of the S operend are loeded from the R-input port into the most-significent half of the S register, and the least-significent 32 bits of the S operand ere loaded from the S-input port into tha leest-significant helf of the S ragister.

At the same time, et 2, tha output of the R-tamp register is loeded into tha most-significant half of the R registar, and tha output of the S-temp register is loaded into the least-significant helf of tha R register.

If tha R-operand is a singla-precision (32-bit) floating-point operand, it is input in tha R-MSW position (with S/D̄R = HIGH) and the data in the R-LSW position is ignored by tha processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/D̄R = HIGH) end the data in the R-MSW position is ignored by the processor. The same procedure epplias to the S-operend, if it is a single-precision operand.

64-Bit Bus, Single-Cycle, S First (M16 = 1, M15 = 0, M14 = 1)

In this mode, the MSW of the 64-bit S operend is pieced on the R-port end the LSW on the S-port. Both helfwords are loaded in the first half cycle. Similarly, the two halves of the R operend are loaded in the second helf cycle. Aftar one full cycle, the R and S registers contain the R and S operands, raspectively.



Timing of Operations with Input Mode 6 (64-Bit Bus, Single-Cycle, S First)*

*Assumes processor flow-through moda, F register, and status register clocked.

The tamporary ragistars ara clocked on avary HIGH-to-LOW clock transition.

At 1, the most-significent 32 bits of the S operand are loaded from the R-input port into the R-temp ragistar, and the laast-significent 32 bits of the S operand are loaded from the S-input port into tha S-temp register.

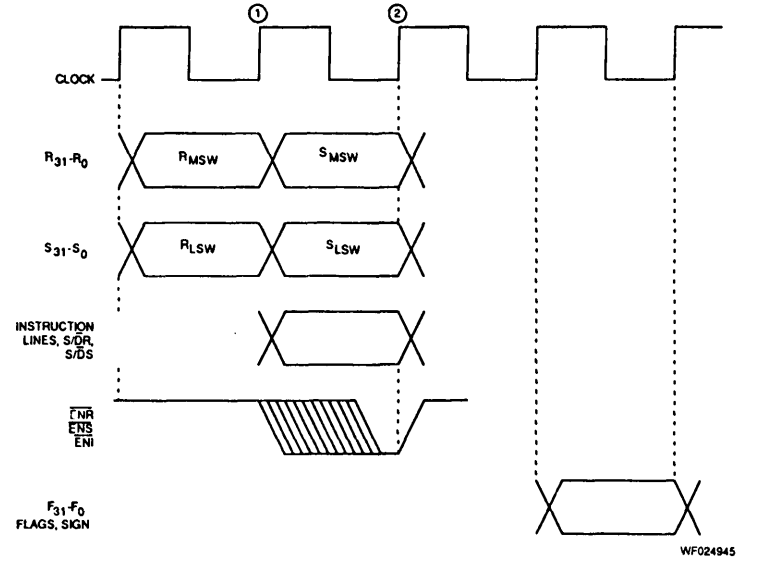
At 2, the most-significant 32 bits of the R oparand are loaded from the R-input port into the most-significant half of tha R register, and the least-significant 32 bits of the R operand ara loaded from the S-input port into the least-significant half of the R registar.

At the same time, at 2, the output of the R-temp register is loaded into the most-significent half of the S register, and the output of the S-temp register is loeded into tha least-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/$\overline{DR}$ = HIGH) and the data in the R-LSW position is ignored by the procassor. If the R-operand is a single-pracision (32-bit) integar operand, it is input in the R-LSW position (with S/$\overline{DR}$ = HIGH) and the dete in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a singla-precision operand.

64-Bit Bus, Double-Cycle, R First (M16 = 1, M15 = 1, M14 = 0)

In this mode, the MSW of the 64-bit R operend is pieced on the R-port end tha LSW on the S-port. Both halfwords are loeded in the first cycle. Similarly, the two halves of the S operend ere loaded in the second cycle. After two cycles, the R and S registers contain the R and S operands, respectrvely.



Timing of Operations with Input Mode 7 (64-Bit Bus, Double-Cycle, R First)*

*Assumes processor flow-through mode, F register, and status registar clocked.

The temporary registers are clocked on evary LOW-to-HIGH clock transition.

At 1, tha most-significant 32 bits of the R operand ere loaded from tha R-input port into tha R-temp registar, and the leest-significant 32 bits of the R operand ere loeded from the S-input port into the S-temp register.

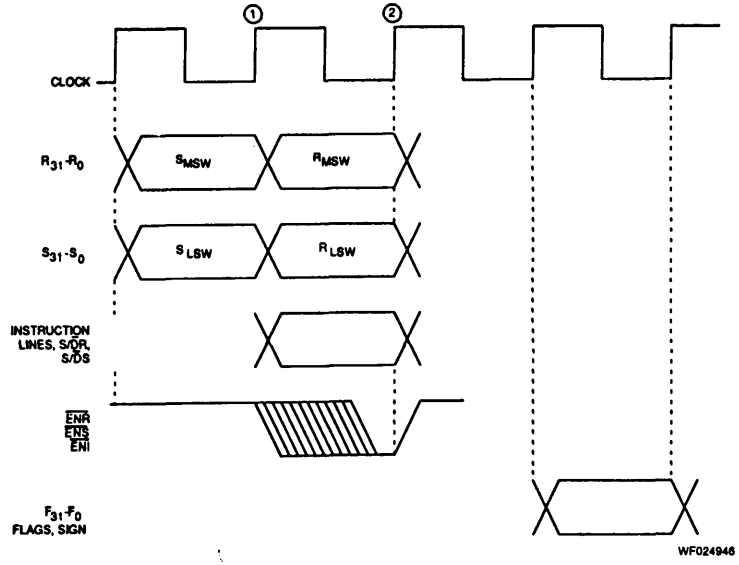
At 2, the most-significant 32 bits of the S operand are loeded from the R-input port into the most-significent helf of the S registar, and the laast-significant 32 bits of the S operand are loaded from tha S-input port into tha least-significant half of the S ragister.

At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the R register, and the output of the S-temp registar is loaded into the laast-significant half of tha R register.

If the R-operend is a single-precision (32-bit) floating-point operend, it is input in the R-MSW position (with S/$\overline{DR}$ = HIGH) end the data in the R-LSW position is ignored by tha processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/$\overline{DR}$ = HIGH) and the data in tha R-MSW position is ignored by the processor. The same procedure applias to the S-operand, if it is e single-precision operend.

64-Bit Bus, Double-Cycle, S First (M16 = 1, M15 = 1, M14 = 1)

In this mode, the MSW of the 64-bit S operand is placed on the R-port and the LSW on the S-port. Both halfwords are loaded in the first cycle. Similarly, the two halves of the R operand are loaded in the second cycle. After two cycles, the R and S registers contain the R and S operands, respectively.



Timing of Operations with Input Mode 8 (64-Bit Bus, Double-Cycle, S First)*

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every LOW-to-HIGH clock transition.

At 1, the most-significant 32 bits of the S operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register.

At 2, the most-significant 32 bits of the R operand are loaded from the R-input port into the most-significant half of the R register, and the least-significant 32 bits of the R operand are loaded from the S-input port into the least-significant half of the R register.

At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the S register, and the output of the S-temp register is loaded into the least-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/$\overline{D}$R = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/$\overline{D}$R = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

Pipelining of Operations

The floating-point ALU of the Am29C327 may be operated in one of three modes:

1. Flow-Through Mode
2. Single-Pipelined Mode
3. Double-Pipelined Mode

Flow-Through Mode

In this mode the floating-point ALU acts as a purely combinatorial device.

Single-Pipelined Mode

In this mode the floating-point ALU contains a single pipeline delay for all operations; throughput is roughly double that for unpipelined mode. Simplified diagrams of the ALU configuration for single-pipelined mode are shown in Figure 2.

Double-Pipelined Mode

In this mode, which applies only to the multiply-accumulate operation, the ALU contains two pipeline delays; throughput is roughly triple that for the unpipelined multiplication-accumulation operation. A simplified block diagram is shown in Figure 3.

Figures 4 and 5 provide timing diagrams for all operations except multiply-accumulate, illustrating flow-through mode and pipelined mode, respectively. Figures 6, 7, and 8 provide timing diagrams for multiply-accumulate, illustrating flow-through mode, single-pipelined mode, and double-pipelined mode, respectively.

The choice of pipelining mode affects only the floating-point ALU. Operations of other parts of the Am29C327, such as the input registers, the output register, the mode register, and the instruction register are not affected by the choice of pipelining mode. However, the instruction bits are pipelined as they pass through the ALU. This permits instructions to be interleaved in pipelined mode.

The desired pipelined mode or modes can be invoked by setting mode register bits M20 and M19 to the appropriate values.

When using the Am29C327 in either single-pipelined or double-pipelined mode, two conditions must be observed:

1. The "load mode register" instruction is not pipelined, nor are any of the mode register bits. When the mode register is loaded, any differences between the current mode and the previous mode take effect immediately. In single-pipelined mode, the user should separate the last valid ALU instruction and the "load mode register" instruction with one "NO-OP" instruction. In double-pipelined mode, the user should separate them with two "NO-OP" instructions. A NO-OP instruction is any instruction whose result is not stored in register F, or the register file.
2. A multiplication-accumulation instruction cannot be immediately followed by any other type of instruction. This problem can be avoided by inserting a "dummy" multiplication-accumulation instruction at the end of a multiplication-accumulation instruction. This "dummy" instruction is any instruction whose results are not stored in register F or the register file.

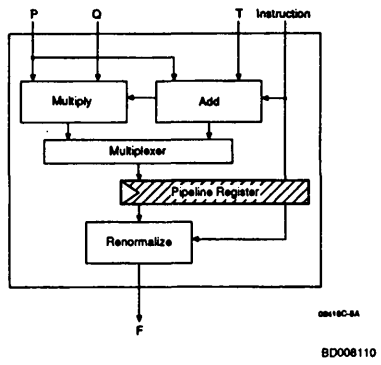


Figure 2.1. ALU in Pipelined Mode — All Operations Except Multiply-Accumulate

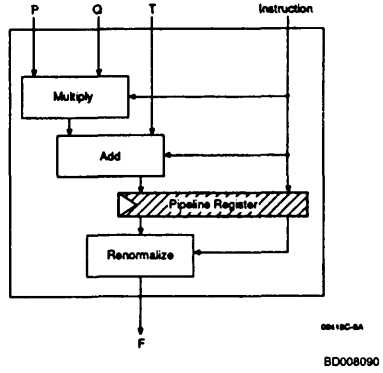
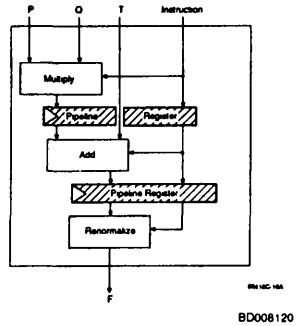
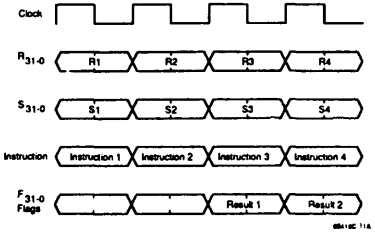


Figure 2.2. ALU in Single-Pipelined Mode — Multiply-Accumulate Only



BD008120

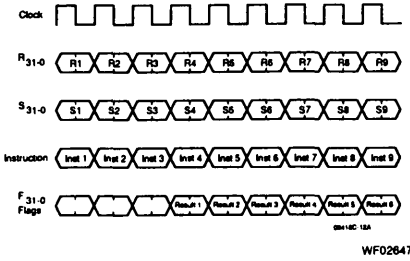
Figure 3. ALU Double-Pipelined Mode — Multiply-Accumulate Only



WF026460

Figure 4. Flowthrough Mode — All Operations Except Multiply-Accumulate*

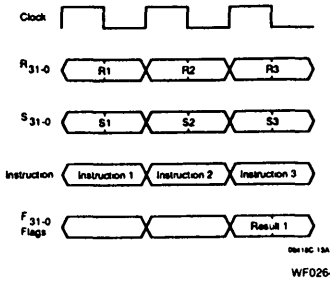
*Assumes F-register and status register clocked



WF026470

Figure 5. Pipelined Mode — All Operations Except Multiply-Accumulate*

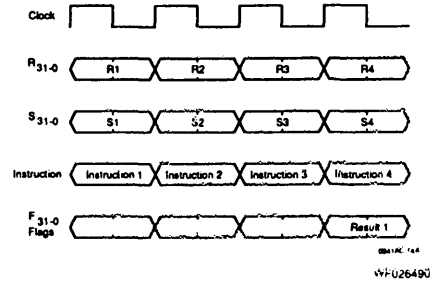
*Assumes F-register and status register clocked



WF026460

Figure 6. Flow-Through Mode — Multiply-Accumulate Operations Only*

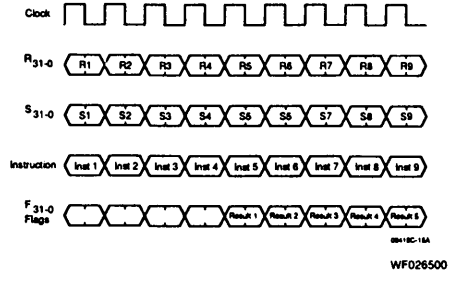
*Assumes F-register and status register clocked.



WF026490

Figure 7. Single-Pipelined Mode — Multiply-Accumulate Operations Only*

*Assumes F-register and status register clocked.



WF026500

Figure 8. Double-Pipelined Mode — Multiply-Accumulate Operations Only*

*Assumes F-register and status register clocked.

Instruction Set

Instruction Register Format

The 14-bit instruction word $I_{13} - I_0$ comprises the sign-change controls, the integer/floating-point select bit, and the opcode.

I_{13}	I_{12} I_{11}	I_{10} I_9	I_8 I_7	I_6	I_5	I_4 I_3 I_2 I_1 I_0
SIGN (P)	SIGN (Q)	SIGN (T)	SIGN (F)	INT/FP	OPCODE	

The opcode field, $I_4 - I_0$, specifies the base operation to be performed by the ALU; instruction bit I_5 selects between floating-point and integer operations. The base operations and their corresponding opcodes are listed in Table 1.

TABLE 1. BASE OPERATIONS PERFORMED BY THE Am29C327

I5-0 5	I5-0 4 3 2 1 0	Mnemonic	Base Operation (Floating-Point)
0	0 0 0 0 0	FPAS	F' = P
0	0 0 0 0 1	FADD	F' = P' + T'
0	0 0 0 1 0	FMUL	F' = P' * Q'
0	0 0 0 1 1	FCMP	Compare P, T
0	0 0 1 0 0	FMAX	Maximum P, T
0	0 0 1 0 1	FMIN	Minimum P, T
0	0 0 1 1 0	FCTI	Convert T to Integer
0	0 0 1 1 1	FSTI	Scale T to Integer by Q
0	0 1 0 0 0	FMAC	F' = (P' * Q') + T'
0	0 1 0 0 1	FRND	Round T to Integral Value
0	0 1 0 1 0	FRCP	Reciprocal Seed of P
0	0 1 0 1 1	FCTA	Convert T to Alternate FP Format
0	0 1 1 0 0	FCFA	Convert T from Alternate FP Format
I5-0 5	**I5-0 4 3 2 1 0**	**Mnemonic**	**Base Operation (Integer)**
1	0 0 0 0 0	IPAS	F = P
1	0 0 0 0 1	IADD	F = P + T
1	0 0 0 1 0	IMUL	F = P * Q
1	0 0 0 1 1	ICMP	Compare P, T
1	0 0 1 0 0	IMAX	Maximum P, T
1	0 0 1 0 1	IMIN	Minimum P, T
1	0 0 1 1 0	ICTF	Convert T to Floating-Point
1	0 0 1 1 1	ISTF	Scale T to Floating-Point by Q
1	1 0 0 0 0	ILOR	F = P OR T
1	1 0 0 0 1	IAND	F = P AND T
1	1 0 0 1 0	IXOR	F = P XOR T
1	1 0 0 1 1	ILSH	Shift P Logical by Q Places
1	1 0 1 0 0	IASH	Shift P Arithmetic by Q Places
1	1 0 1 0 1	IFSH	Funnel Shift PT by Q Places
I5-0 5	**I5-0 4 3 2 1 0**	**Mnemonic**	**Base Operation (Format-Independent)**
X	1 1 0 0 0	MOVE	Move P
X	1 1 1 1 1	LMRG	Load Mode Register

Sign-Change Selects

Each ALU input and output operand has associated hardware that can be used to modify operand signs (see Figure 9). These sign-change blocks, when applied to base operations, greatly increase the number of available operations. A base operation of P' + T', for example, can be used to perform operations such as P - T, ABS(P + T), ABS(P) + ABS(T), and others, simply by modifying the signs of the input and output operands.

Using the sign-change blocks, the sign of an input operand may be left unchanged, complemented, set to zero, or set to one; the sign of the output operand may be left unchanged, complemented, set to zero, set to one, set to the sign of the P input operand, or set to the sign of the T input operand. Select decodes for the P, Q, T, and F operand sign-change blocks are shown in Table 2-1, 2-2, 2-3, and 2-4, respectively.

Note: The P-sign change block has no effect on the P input operand for the base operation F' = P.

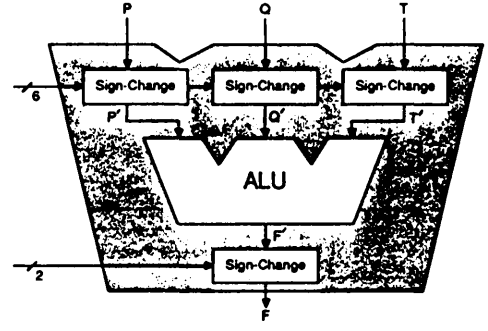


Figure 9. ALU Sign-Change Blocks

TABLE 2-1. SELECT DECODE FOR P OPERAND SIGN-CHANGE BLOCK

I_{13}	I_{12}	Sign (P')
0	0	SIGN (P)
0	1	$\overline{\text{SIGN}}$ (P)
1	0	0
1	1	1

Note: The P-sign change block has no effect on the P input operand for the base operation F' = P.

TABLE 2-2. SELECT DECODE FOR Q OPERAND SIGN-CHANGE BLOCK

I_{11}	I_{10}	Sign (Q')
0	0	SIGN (Q)
0	1	$\overline{\text{SIGN}}$ (Q)
1	0	0
1	1	1

TABLE 2-3. SELECT DECODE FOR T OPERAND SIGN-CHANGE BLOCK

I_9	I_8	Sign (T')
0	0	SIGN T
0	1	$\overline{\text{SIGN}}$ T
1	0	0
1	1	1

TABLE 2-4. SELECT DECODE FOR F OPERAND SIGN-CHANGE BLOCK

Base Operation	I_{11}	I_{10}	I_7	I_6	Sign (F)
F' = P	0	x	0	0	SIGN (F')
or	0	x	0	1	$\overline{\text{SIGN}}$ (F')
Maximum P, T	0	x	1	0	0
or	0	x	1	1	1
Minimum P, T	1	0	x	x	SIGN (P)
	1	1	x	x	SIGN (T)
	x	x	0	0	SIGN (F')
All other	x	x	0	1	$\overline{\text{SIGN}}$ (F')
base operations	x	x	1	0	0
	x	x	1	1	1

Operand Multiplexer Selects

The instruction fields $PSEL_3$ - $PSEL_0$, $QSEL_3$ - $QSEL_0$, and $TSEL_3$ - $TSEL_0$ specify the select codes for the P, Q, and T operand multiplexers, respectively; the codes are summarized in Table 3.

TABLE 3. OPERAND MULTIPLEXER SELECT CODES

$PSEL_3$ $QSEL_3$ $TSEL_3$	$PSEL_2$ $QSEL_2$ $TSEL_2$	$PSEL_1$ $QSEL_1$ $TSEL_1$	$PSEL_0$ $QSEL_0$ $TSEL_0$	P Q T
0	0	0	0	R - Register
0	0	0	1	S - Register
0	0	1	0	0
0	0	1	1	0.5 (Floating Point); -1 (Integer)
0	1	0	0	1
0	1	0	1	2
0	1	1	0	3
0	1	1	1	Pi (Floating Point); Max Neg. Value (Integer)
1	0	0	0	Register File 0 (RF0)
1	0	0	1	Register File 1 (RF1)
1	0	1	0	Register File 2 (RF2)
1	0	1	1	Register File 3 (RF3)
1	1	0	0	Register File 4 (RF4)
1	1	0	1	Register File 5 (RF5)
1	1	1	0	Register File 6 (RF6)
1	1	1	1	Register File 7 (RF7)

Operand Precisions

The Am29C327 supports mixed-precision operations, so that it is possible, for example, for an operation to have single-precision inputs and a double-precision output, or one single- and one double-precision input, or any other combination.

Precision of the operands in registers R and S is specified by signals S/$\overline{DR}$ and S/$\overline{DS}$ at the time the operands are loaded.

The precision of an operation result is specified by signal S/$\overline{DF}$.

Operands stored in the register file are each accompanied by a bit indicating that operand's precision; this precision information is automatically supplied to the ALU when a register file location is used as an input operand to an operation.

Processor Operations

Table 4 illustrates a number of possible ALU instructions comprising the opcode, integer/floating-point select, and sign-change fields. Note that the remaining instruction bits — P, Q, and T operand multiplexer selects; the rounding modes; and the output operand precision — can be specified independently.

The user may create instructions using instruction words other than those listed in Table 4. For some base operations, sign-change control settings are completely arbitrary; for others, only the sign-change field values shown in Table 4 are valid. Table 5 summarizes permissible sign-change field values for each core operation.

TABLE 4. INSTRUCTION WORDS

Operation	Sign P	Sign Q	Sign T	Sign F	I/$\overline{F}$	Opcode
FP F = P	xx	0x	xx	00	0	00000
FP F = -P	xx	0x	xx	01	0	00000
FP F = ABS (P)	xx	0x	xx	10	0	00000
FP F = Sign (T)*ABS (P)	xx	11	xx	xx	0	00000
FP F = P + T	00	xx	00	00	0	00001
FP F = P - T	00	xx	01	00	0	00001
FP F = T - P	01	xx	00	00	0	00001
FP F = -P - T	01	xx	01	00	0	00001
FP F = ABS (P + T)	00	xx	00	10	0	00001
FP F = ABS (P - T)	00	xx	01	10	0	00001
FP F = ABS (P) + ABS (T)	10	xx	10	00	0	00001
FP F = ABS (P) - ABS (T)	10	xx	11	00	0	00001
FP F = ABS (ABS (P) - ABS (T))	10	xx	11	10	0	00001
FP F = P * Q	00	00	xx	00	0	00010
FP F = (-P) * Q	01	00	xx	00	0	00010
FP F = ABS (P * Q)	00	00	xx	10	0	00010
FP Compare P, T	00	xx	01	00	0	00011
FP Max P, T	00	00	01	00	0	00100
FP Max ABS (P), ABS (T)	10	00	11	00	0	00100
FP Min P, T	01	00	00	00	0	00101
FP Min ABS (P), ABS (T)	11	00	10	00	0	00101
FP Limit P to Magnitude T	11	10	xx	xx	0	00101
FP Convert T to Integer	xx	xx	00	00	0	00110
FP Scale T to Integer by Q	xx	00	00	00	0	00111
FP F = (P*Q) + T	00	00	00	00	0	01000
FP F = T - (P*Q)	01	00	00	00	0	01000
FP F = (P*Q)-T	00	00	01	00	0	01000
FP F = -T - P*Q	01	00	01	00	0	01000
FP F = ABS (T) + ABS (P*Q)	10	10	10	00	0	01000
FP F = ABS (T) - ABS (P*Q)	11	10	10	00	0	01000
FP F = ABS (P*Q) - ABS (T)	10	10	11	00	0	01000
FP Round T to Integral Value	xx	xx	00	00	0	01001
FP Reciprocal Seed (P)	00	xx	xx	00	0	01010
FP Convert T to Alternate Floating-point Formet	xx	xx	00	00	0	01011
FP Convert T from Alternate Floating-point Format	xx	xx	00	00	0	01100
Int F = P	00	00	00	00	1	00000
Int F = -P	00	00	00	01	1	00000
Int F = ABS (P)	00	00	00	10	1	00000
Int F = sign (T)*ABS (P)	00	11	00	xx	1	00000
Int F = P + T	00	xx	00	00	1	00001
Int F = P - T	00	xx	01	00	1	00001
Int F = T - P	01	xx	00	00	1	00001
Int F = ABS (P + T)	00	xx	00	10	1	00001
Int F = ABS (P - T)	00	xx	01	10	1	00001
Int F = P * Q	00	00	xx	00	1	00010
Int Compare P, T	00	xx	01	00	1	00011
Int Max P, T	00	00	01	00	1	00100
Int Min P, T	01	00	00	00	1	00101

TABLE 4. INSTRUCTION WORDS (Cont'd.)

Operation	Sign P	Sign Q	Sign T	Sign F	I/F	Opcode
Int **Convert T to Float**	xx	xx	00	00	1	00110
Int **Scale T to Float by Q**	xx	00	00	00	1	00111
Int **P OR T**	xx	xx	xx	xx	1	10000
Int **P AND T**	xx	xx	xx	xx	1	10001
Int **P XOR T**	xx	xx	xx	xx	1	10010
Int **NOT T (see Note 1)**	xx	xx	xx	xx	1	10010
Int **Shift P Logical by Q Places**	00	00	xx	00	1	10011
Int **Shift P Arithmetic by Q Places**	00	00	xx	00	1	10100
Int **Funnel Shift PT by Q Places**	00	00	00	00	1	10101
Move P	xx	xx	xx	xx	x	11000
Load Mode Register	xx	xx	xx	xx	x	11111

Notes: 1. NOT T is performed by XORing T with a word containing all 1's (integer – 1). When involving NOT T the user must set $PSEL_3$ – $PSEL_0$ to 0011_2, thus selecting integer constant – 1.

TABLE 5. ALLOWABLE SIGN-CHANGE/BASE OPERATION COMBINATIONS

	I IIIII 5 43210	Base Operation	Sign-Change Fields Sign (P)	Sign (Q)	Sign (T)	Sign (F)
Floating-Point Operations	0 00000	FP F' = P*	V	V	x	V
	0 00001	FP F' = P' + T'	V	x	V	V
	0 00010	FP F' = P'*Q'	V	V	x	V
	0 00011	FP Compara P, T	F	x	F	F
	0 00100	FP Max P, T	F	F	F	F
	0 00101	FP Min P, T	F	F	F	F
	0 00110	FP Cvt T to Int	x	x	F	F
	0 00111	FP Scala T to Int	x	F	F	F
	0 01000	FP F' = (P'*Q') + T'	V	V	V	V
	0 01001	FP Round T	x	x	F	F
	0 01010	FP Recip Seed P	F	x	x	F
	0 01011	FP Cvt T to Alt Fmt	x	x	F	F
	0 01100	FP Cvt T fm Alt Fmt	x	x	F	F
Integer Operations	1 00000	Int F = P	F	F	F	F
	1 00001	Int F = P + T	F	x	F	F
	1 00010	Int F = P*Q	F	F	x	F
	1 00011	Int Compare P, T	F	x	F	F
	1 00100	Int Max P, T	F	F	F	F
	1 00101	Int Min P, T	F	F	F	F
	1 00110	Int Cvt T to f.p.	x	x	F	F
	1 00111	Int Scala T to f.p.	x	F	F	F
	1 10000	Int F = P OR T	x	x	x	x
	1 10001	Int F = P AND T	x	x	x	x
	1 10010	Int F = P XOR T	x	x	x	x
	1 10011	Int Shift P Logical	F	F	x	F
	1 10100	Int Shift P Arith	F	F	x	F
	1 10101	Int Funnal Shift PT	F	F	F	F
	x 11000	Mova P	x	x	x	x
	x 11111	Load Moda Reg	x	x	x	x

Key: V = Variable; user can specify arbitrary sign change.
F = Fixed; user is restricted to sign change combinations shown in Table 4.
x = Don't care; this field does not affect the operation or its result.

*Nota: Tha P-sign changa block has no affect on tha P-input operand for the base operation F' = P.

Base Operation Code Description

F' = P (Floating-point) FPAS: The P-operand is passed through the ALU unchanged, except for any specified precision conversions. If the usar specifies diffarant input and output precisions, the operation may ba used to perform single-to-doubla or doubla-to-singla conversions. Instructions such as negation, absolute valua extraction and sign-transfer may be axecuted by setting tha sign-change controls appropriataly while axecuting this base operation.

*Note: Tha P-sign change block has no affect on the P-input operand for tha base operation F' = P.

F' = P' + T' (Floating-point) FADD: The two operands P' and T' ara added, taking into account any specified precision conversions. Instructions such as subtraction, sum-of-absolute-valuas, differanca-of-absoluta-values, absolute-value-of-sum, and absoluta-value-of-diffarance may be axecuted by satting tha sign-change controls appropriataly whila axecuting this basa operation.

F' = P' * Q' (Floating-point) FMUL: The Operands P' and Q' ara multiplied, taking into account any specified precision conversions. Instructions such as negativa-product and absoluta-valua-of-product may be axacuted by setting tha sign-changa controls appropriataly while axecuting this base operation.

Compare P, T (Floating-point) FCMP: The two operands P and T ara compared, taking into account any specified precision conversions. Tha output of the operation is tha rasult of tha subtraction (P-T). Tha flags are set appropriately to indicata the rasult of tha comparison, conforming to the relavant parts of tha floating-point standards. For IEEE and DEC operations, ona of four flags (graater than, less than, equals or unordered) is set for any givan compara operation. For IBM operations, the unordared flag does not apply sinca tha format does not support any reserved operands.

Maximum P, T (Floating-point) FMAX: The two operands P and T are comparad, taking into account any specified precision conversions. Tha most-positive operand is selected as tha output. Tha "Winner" flag indicates which of the operands is selected. Additionally, the operation maximum-of-absoluta-value may be performed by satting tha appropriata sign-change controls.

Minimum P, T (Floating-point) FMIN: The two operands P and T ara compared, taking into account any specified precision conversions. Tha most-negativa operand is selacted as tha output. Tha "Winner" flag indicatas which of the two operands is selected. Additionally, tha operations minimum-of-absoluta-valuas and limit-P-to-magnitude-T may be performed by satting tha appropriate sign-changa controls. Tha limit-P-to-magnituda-T operation is useful for "clipping" a sequenca of operands to ansura that thair magnituda nevar exceeds a preset limit.

Convert T to Integer (Floating-point) FCTI: Tha operand T is convarted from floating-point raprasentation to two's complamant intager raprasantation, taking into account tha specified precision of the floating-point operand. If tha output precision is specified as singla, the result is a 32-bit integer. If tha output precision is specified as doubla, tha result is a 64-bit integer.

Scale T to Integer by Q (Floating-point) FSTI: The operand T is convarted from floating-point raprasentation to two's complamant integer raprasantation, using tha axponent of tha floating-point operand Q as a scala factor and taking into account the specified precision of tha floating-point operands. The unbiased axponant of the operand Q is added to tha axponant of tha operand T, permitting IEEE and DEC operands to be multiplied by any power of 2 and IBM operands by any power of 16, before tha convarsion is performed. If the output precision is specified as singla, the result is a 32-bit integar. If the output precision is specified as doubla, tha rasult is a 64-bit integer.

F' = (P' * Q') + T' (Floating-point) FMAC: The operands P' and Q' ara multiplied, producing a double-precision product. This product is added to the operand T', taking into account any specified precision conversions. Instructions such as "P*Q – T'", "T – P*Q'", "ABS(P*Q) + ABS(T)" and "ABS (P*Q + T)" may be executed by setting the sign-changa controls appropriately whila axacuting this base operation.

Round T to Integral Value (Floating-point) FRND: The floating-point operand T is rounded to an integer-valued floating-point operand, using the specified rounding mode and taking into account any specified precision conversions. As an example, the operation converts a floating-point representation of Pi (3.14159...) to a floating-point reprasantation of 3.0 or 4.0, dapending on tha rounding mode selected. The final result of the operation is a floating-point number.

Reciprocal Seed of P (Floating-point) FRCP: An approximation to the reciprocal of the operand P is evaluated, taking into account any specifiad precision convarsions. The reciprocal seed comprises an accurata sign, a fully-accurata axponent and a mantissa which is accurata to only one placa. This operation can be used as tha initial stap in performing Newton-Raphson division; optionally, an external seed look-up table can be used for faster convargance.

Convert T to Alternate Floating-point Format (Floating-point) FCTA: Tha floating-point operand T, assumed to be in tha "primary" floating-point format is converted to a floating-point operand in tha "Altarnata" floating-point format, taking into account any specified precision conversions.

Convert T from Alternate Floating-point Format (Floating-point) FCFA: Tha floating-point operand T, assumed to be in tha "Alternata" floating-point format is converted to a floating-point operand in the "primary" floating-point format, taking into account any specified precision convarsions.

F = P (Integer) IPAS: Tha P-operand is passed through the ALU unchanged, axcapt for any specified precision convarsions. If the user specifies different input and output precisions, tha operation may be used to perform singla-to-doubla or doubla-to-singla conversions. Instructions such as negation, absoluta value axtraction and sign transfar may be performed by setting tha sign-changa control appropriataly whila axecuting this base operation.

F = P + T (Integer) IADD: Tha two operands P and T ara added, taking into account any specified precision convarsions. Instructions such as subtraction, absoluta-valua-of-sum and absoluta-value-of-difference may ba performed by setting the sign-changa controls appropriataly whila axecuting this base operation.

F = P * Q (Integer) IMUL: The two operands P and Q are multiplied, taking into account any specified precision conversions. Eithar 32-bit multiplication or 64-bit multiplication may be performed, and the user may select aither tha MSBs or the LSBs of the product as tha final rasult. In addition, format-adjusting may be implamented if required, and the operands may be considered as signed (two's complament) or unsigned.

Compare P, T (Integer) ICMP: Tha two operands P and T are compared, taking into account any specified precision conversions. The output of the operation is tha rasult of tha subtraction (P - T). Tha flags ara sat appropriataly to indicata the rasult of the comparison, one of three flags (graatar than,

less than or equals) being set for eny given compere operetion.

Maximum P, T (Integer) IMAX: The two operends P end T ere compered, taking into account eny specified precision conversions. The most-positive operand is selected es the output. The "Winner" fleg indicetes which of the two operends is selected.

Minimum P, T (Integer) IMIN: The two operends P end T ere compered, taking into account eny specified precision conversions. The most-negetive operend is selected es the output. The "Winner" fleg indicates which of the two operands is selected.

Convert T to Floating-point (Integer) ICTF: The operend T is converted from two's complement integer representetion to floeting-point representation, taking into account the specified precision of the integer operend. If the output precision is specified as single, the result is e 32-bit floeting-point operand. If the output precision is specified es double, the result is e 64-bit floeting-point operend.

Scale T to Floating-point by Q (Integer) ISTF: The operend T is converted from two's complement integer representation to floeting-point representetion, using the exponent of the floeting-point operend Q es e scele fector end teking into eccount the specified precision of the integer operend. The unbiesed exponent of the operend Q is edded to the exponent of the floeting-point result, permitting IEEE end DEC operends to be multiplied by eny power of 2, end IBM operands by eny power of 16, efter the conversion is performed. If the output precision is specified es single, the result is a 32-bit floeting-point operend. If the output precision is specified es double, the result is e 64-bit floeting-point operend.

F = P OR T (Integer) ILOR: The operend P is logicelly ORed with the operand T. Before the operetion is performed, the inputs, if 32-bit, ere sign-extended to 64-bits.

F = P AND T (Integer) IAND: The operend P is logicelly ANDed with the operend T. Before the operetion is performed, the inputs, if 32-bit, ere sign-extended to 64-bits.

F = P XOR T (Integer) IXOR: The operand P is logicelly exclusive-ORed with the operend T. Before the operetion is performed, the inputs, if 32-bit, ere sign-extended to 64-bits. This operetion mey be used to invert an operend by selecting the second operend to be the integer constant -1, so thet ell bits of this second operand ere 1. Exclusive-ORing an operand with -1 is equivalent to inverting eech bit in the operend.

Shift P Logical by Q Places (Integer) ILSH: This operetion cennot be performed in mixed-precision mode. The precision of the result is the same as the precision of the input operand P. A two's-complement shift length in the range -64 to +63 (double-precision) or -32 to +31 (single-precision) is extracted from the LSBs of the operand Q. The operend P is logicelly right-shifted by the number of places specified by the shift length. A negative shift length therefore produces e left-shift. If e right-shift is performed, zeros fill vacated bit positions to the left of the input operend. If e left-shift is performed, zeros fill vecated bit positions to the right of the input operand.

Shift P Arithmetic by Q Places (Integer) IASH: This operetion cennot be performed in mixed-precision mode. The precision of the result is the same es the precision of the input operend P. A two's-complement shift length in the renge -64 to +63 (double-precision) or -32 to +31 (single-precision) is extrected from the LSBs of the operand Q. The operand P is erithmeticelly right-shifted by the number of pleces specified by the shift length. A negetive shift length therefore produces a left-shift. If e right-shift is performed, the MSB (bit 63 or 31) is repliceted to fill veceted bit positions to the left of the input operand. If e left-shift is performed, zeros fill veceted bit positions to the right of the input operend.

Funnel Shift PT by Q Places (Integer) IFSH: This operetion cennot be performed in mixed-precision mode. The operend T is interpreted es heving the same precision as the input operand P and the precision of the result is elso the same es the precision of the input operend P. A two's-complement shift length in the renge -64 to +63 (double-precision) or -32 to +31 (single-precision) is extracted from the LSBs of the operand Q. A triple-width operend (96-bit or 192-bit) is formed by concateneting the input operands into the errangement P-T-P, with the 32-bit or 64-bit result field initially eligned with the T-operand. The triple-width operand is logicelly right-shifted by the number of pleces specified by the shift length. A negative shift length therefore produces e left-shift.

Move P (Format Independent) MOVE: The 64-bit operend P is passed unchenged through the ALU. No exceptions ere detected or signelled.

Load Mode Register (Format Independent) LMRG: The 32-bit operend on the $R_{31:0}$ input port is loeded into the mode register on the rising edge of the clock input. No exceptions ere detected or signelled.

Operation Flags

For eech operation, the ALU produces thirteen flegs that indicete operation stetus. Of the flags produced, e maximum of seven ere relevant to eny given operetion. The relevant flags ere clocked into the stetus register, end the other flegs ere discarded.

The ALU flegs are:

C — CARRY: Carry-out bit produced by integer eddition, subtraction, or comperison.

I — INVALID OPERATION: Input operands are unsuiteble for the operetion specified (e.g., ∞ * 0).

R — RESERVED OPERAND: Reserved operend detected/ genereted.

S — SIGN: Result sign.

U — UNDERFLOW: Result underflowed the destinetion format.

V — OVERFLOW: Result overflowed the destinetion formet.

W — WINNER: Indicates which of the two operands wes selected when performing Max/Min operetions.

X — INEXACT RESULT: Result had to be rounded to fit the destination formet.

Z — ZERO: Zero result.

>, =, <, # — GREATER THAN, EQUAL, LESS THAN, UNORDERED: Used to report the result of e comperison operetion.

Table 6 lists the flegs reported for eech operetion.

TABLE 6. ORGANIZATION OF FLAGS APPLICABLE TO EACH BASE OPERATION

Format	Operation	Mnemonic	SIGN	Flag6	Flag5	Flag4	Flag3	Flag2	Flag1
IEEE	F' = P	FPAS	S	Z	X	U	V	R	I
IEEE	F' = P' + T'	FADD	S	Z	X	U	V	R	I
IEEE	F' = P' x Q'	FMUL	S	Z	X	U	V	R	I
IEEE	Compare P, T	FCMP	S	=	>	<	#	R	I
IEEE	Maximum P, T	FMAX	S	Z		W		R	I
IEEE	Minimum P, T	FMIN	S	Z		W		R	I
IEEE	Convert T to Integer	FCTI	S	Z	X		V	R	I
IEEE	Scale T to Integer	FSTI	S	Z	X		V	R	I
IEEE	F' = (P' x O') + T'	FMAC	S	Z		U	V	R	I
IEEE	Round T to Integral Value	FRND	S	Z	X		V	R	I
IEEE	Reciprocal Seed of P	FRCP	S	Z		U	V	R	I
IEEE	Convert T to Alt FP Format	FCTA	S	Z	X	U	V	R	I
IEEE	Convert T from Alt FP Format	FCFA	S	Z	X	U	V	R	I
DEC D	F' = P	FPAS	S	Z	X		V	R	
DEC D	F' = P' + T'	FADD	S	Z	X	U	V	R	
DEC D	F' = P' x Q'	FMUL	S	Z	X	U	V	R	
DEC D	Compere P, T	FCMP	S	=	>	<	#	R	
DEC D	Maximum P, T	FMAX	S	Z		W		R	
DEC D	Minimum P, T	FMIN	S	Z		W		R	
DEC D	Convert T to Integer	FCTI	S	Z	X		V	R	I
DEC D	Scale T to Integer	FSTI	S	Z	X		V	R	I
DEC D	F' = (P' x O') + T'	FMAC	S	Z		U	V	R	
DEC D	Round T to Integral Value	FRND	S	Z	X		V	R	
DEC D	Reciprocal Seed of P	FRCP	S	Z		U	V	R	I
DEC D	Convert T to Alt FP Format	FCTA	S	Z	X	U	V	R	I
DEC D	Convert T from Alt FP Format	FCFA	S	Z	X	U	V	R	I
DEC G	F' = P	FPAS	S	Z	X	U	V	R	
DEC G	F' = P' + T'	FADD	S	Z	X	U	V	R	
DEC G	F' = P' x Q'	FMUL	S	Z	X	U	V	R	
DEC G	Compere P, T	FCMP	S	=	>	<	#	R	
DEC G	Maximum P, T	FMAX	S	Z		W		R	
DEC G	Minimum P, T	FMIN	S	Z		W		R	
DEC G	Convert T to Integer	FCTI	S	Z	X		V	R	I
DEC G	Scale T to Integer	FSTI	S	Z	X		V	R	I
DEC G	F' = (P' x O') + T'	FMAC	S	Z		U	V	R	
DEC G	Round T to Integral Value	FRND	S	Z	X		V	R	
DEC G	Reciprocal Seed of P	FRCP	S	Z		U	V	R	I
DEC G	Convert T to Alt FP Format	FCTA	S	Z	X	U	V	R	I
DEC G	Convert T from Alt FP Format	FCFA	S	Z	X	U	V	R	I
IBM	F' = P	FPAS	S	Z	X		V		
IBM	F' = P' + T'	FADD	S	Z	X	U	V		
IBM	F' = P' x Q'	FMUL	S	Z	X	U	V		
IBM	Compare P, T	FCMP	S	Z	>	<			
IBM	Maximum P, T	FMAX	S	Z		W			
IBM	Minimum P, T	FMIN	S	Z		W			
IBM	Convert T to Integer	FCTI	S	Z	X		V		
IBM	Scale T to Integer	FSTI	S	Z	X		V		
IBM	F' = (P' x O') + T'	FMAC	S	Z		U	V		
IBM	Round T to Integral Value	FRND	S	Z	X		V		
IBM	Reciprocal Seed of P	FRCP	S	Z			V		I
IBM	Convert T to Alt FP Format	FCTA	S	Z	X	U	V	R	
IBM	Convert T from Alt FP Format	FCFA	S	Z	X	U	V	R	I
Integer	F = P	IPAS	S	Z			V		
Integer	F = P + T	IADD	S	Z			V		C
Integer	F = P x Q	IMUL	S	Z			V		
Integer	Compare P, T	ICMP	S	=	>	<	V		C
Integer	Maximum P, T	IMAX	S	Z		W			
Integer	Minimum P, T	IMIN	S	Z		W			
Integer	Convert T to Floating-Point	ICTF	S	Z	X				
Integer	Scale T to Floating-Point	ISTF	S	Z	X	U	V	R	
Integer	F = P OR T	ILOR	S	Z					
Integer	F = P AND T	IAND	S	Z					
Integer	F = P XOR T	IXOR	S	Z					
Integer	Logical Shift P by Q Places	ILSH	S	Z					
Integer	Arithmetic Shift P by Q Places	IASH	S	Z			V		
Integer	Funnel Shift P T by Q Places	IFSH	S	Z					
	Move P	MOVE	S						
	Load Mode Register	LMRG							

Master/Slave Operation

Two Am29C327 processors can be tied together in master/slave configuration, with the slave checking the results produced by the master. All input and output signals of the slave, with the exception of $\overline{SLAVE}$ and MSERR, are tied to the corresponding signals of the master. The master is selected by asserting signal $\overline{SLAVE}$ LOW; the slave, by asserting signal $\overline{SLAVE}$ HIGH.

The slave processor, by comparing its outputs to the outputs of the master processor, performs a comprehensive check of the operation of the master processor. In addition, the slave processor may detect open circuits and other faults in the electrical path between the master processor and the system. Note that the master processor still performs the comparison between its outputs and its own internally generated results, and is therefore able to detect faults in its output drivers.

APPLICATIONS

Suggestions for Power and Ground Pin Connections

The Am29C327operates in an environment of fast signal rise times and substantial switching currents. Therefore, care must be exercised during circuit board design and layout, as with any high-performance component. The following is a suggested layout, but since systems vary widely in electrical configuration, an empirical evaluation of the intended layout is recommended.

The V_{CCO} and GNDO pins carry output driver switching currents and can be electrically noisy. The V_{CC} and GND pins, which supply the logic core of the device, tend to produce less noise, and the circuits they supply may be adversely affected by noise spikes on the V_{CC} plane. For this reason, it is best to provide isolation between the V_{CC} and V_{CCO} pins, as well as independent decoupling for each. Isolating the GND and GNDO pins is not required.

Printed Circuit-Board Layout Suggestions

1. Use of a multi-layer PC board with separate power, ground, and signal planes is highly recommended.

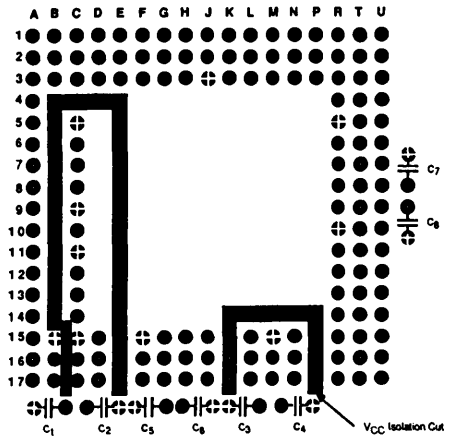
2. All V_{CC} and V_{CCO} pins should be connected to the V_{CC} plane. V_{CC} pins should be isolated from V_{CCO} pins by means of an isolation slot which is cut in the V_{CC} plane; see Figure 10. By physically separating the V_{CC} and V_{CCO} pins, coupled noise will be reduced.

3. All GND and GNDO pins should be connected directly to the ground plane.

4. The V_{CCO} pins should be decoupled to ground with a 0.1-μF ceramic capacitor and a 10-μF electrolytic capacitor, placed as closely to the Am29C327 as is practical. V_{CC} pins should be decoupled to ground in a similar manner.

A suggested layout is shown in Figure 10.

(Bottom View)



● = Through Hole

⊕ = V_{CC} Plane Connection

$C_1 = C_3 = C_5 = C_7 = 0.1\,\mu F$ (ceramic or monolithic capacitor)

$C_2 = C_4 = C_6 = C_8 = 10\,\mu F$ (electrolytic or tantalum capacitor)

CD011712

Figure 10. Suggested Printed Circuit-Board Layout

ABSOLUTE MAXIMUM RATINGS

Storage Temperature ..... −65 to +150°C
Case Temperature Under Bias ..... −55 to +125°C
Supply Voltage to
Ground Potential Continuous ..... −0.3 V to +7.0 V
DC Voltage Applied to Outputs for
HIGH Output State ..... −0.3 V to +V_{CC} +0.3 V
DC Input Voltage ..... −0.3 V to +V_{CC} +0.3 V
DC Output Current, Into LOW Outputs ..... 30 mA
DC Input Current ..... −10 mA to +10 mA

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices
Case Temperature (T_C) ..... 0 to +70°C
Supply Voltage (V_{CC}) ..... +4.75 V to +5.25 V

Military* (M) Devices
Case Temperature (T_C) ..... −55 to +125°C
Supply Voltage (V_{CC}) ..... +4.5 V to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

*Military Product 100% tested at T_C = +25°C, +125°C, and −55°C.

Thermal Resistance (Typical)

Symbol	CGX169	Unit
θ_{JA}	20	°C/W
θ_{JC}	4	°C/W

DC CHARACTERISTICS over operating ranges unless otherwise specified (for APL Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted)

Parameter Symbol	Parameter Description	Test Conditions (Note 1)			Min.	Max.	Unit
V_{OH}	Output HIGH Voltage	V_{CC} = Min., V_{IN} = V_{IH} or V_{IL}	I_{OH} = −4.0 mA		2.4		V
V_{OL}	Output LOW Voltage	V_{CC} = Min., V_{IN} = V_{IH} or V_{IL}	I_{OL} = 4.0 mA			0.5	V
V_{IH}	Input Logical HIGH Voltage (Note 2)				2.0		V
V_{IL}	Input Logical LOW Voltage (Note 2)					0.8	V
$V_{IH}(F)$	Guaranteed Input Logical HIGH Voltage (Note 2)	F Bus, Slave Operation only			V_{CC} − 0.5		V
$V_{IL}(F)$	Guaranteed Input Logical LOW Voltage (Note 2)	F Bus, Slave Operation only				0.5	V
I_{IL}	Input LOW Current	V_{CC} = Max., V_{IN} = 0.5 V				−10	μA
I_{IH}	Input HIGH Current	V_{CC} = Max., V_{IN} = V_{CC} − 0.5 V				10	μA
I_{OZH}	Off State (High Impedance) Output Current	V_{CC} = Max., V_O = 2.4 V				10	μA
I_{OZL}	Off State (High Impedance) Output Current	V_{CC} = Max., V_O = 0.5 V				−10	μA
I_{CC} Static	Static Power Supply Current	V_{CC} = Max., I_O = 0 μA	COM'L T_C = 0 to +70°C	(Note 3) CMOS V_{IN} = V_{CC} or GND		240	mA
				(Note 3) TTL V_{IN} = 0.5 V or 2.4 V		275	
			MIL T_C = −55 to +125°C	(Note 3) CMOS V_{IN} = V_{CC} or GND		285	
				(Note 3) TTL V_{IN} = 0.5 V or 2.4 V		335	
C_{PD}	Power Dissipation Capacitance (Note 4)	V_{CC} = Max., No Load				12,500	pF

Notes: 1. V_{CC} conditions shown as Min. or Max. refer to the applicable device type Operating Range.
2. These input levels provide zero-noise immunity and should only be statically tested in a noise-free environment (not functionally tested).
3. Use CMOS Static I_{CC} when the device is driven by CMOS circuits and TTL Static I_{CC} when the device is driven by TTL circuits.
4. C_{PD} determines the dynamic current consumption:

I_{CC} (Total) = I_{CC} (Static) + $(C_{PD} + nC_L)\frac{f}{2}$, where f is the clock frequency, C_L output load capacitance, and n number of loads.
This is tested on a sample basis.

CAPACITANCE*

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Units
C_I	Input Capacitance	f = 1 MHz		12	pF
C_O	Output Capacitance			20	
$C_{I/O}$	I/O Pin Capacitance			20	

*These capacitances are tested on a sample basis

SWITCHING CHARACTERISTICS over COMMERCIAL operating operating range unless otherwise specified

No.	Parameter Description	Test Conditions	Am29C327		Am29C327-1		Am29C327-2		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	
1	CLOCK Period	(Note 1)							
	Flow-Through Mode								
	Multiply-Accumulate		240	DC	200	DC	160	DC	ns
	All Other Operations		190	DC	180	DC	130	DC	ns
	Single-Pipelined Mode								
	Multiply-Accumulate		180	DC	180	DC	130	DC	ns
	All Other Operations		125	DC	100	DC	80	DC	ns
	Double-Pipelined Mode								
	Multiply-Accumulate		125	DC	100	DC	80	DC	ns
2	CLOCK LOW Time		15		12		10		ns
3	CLOCK HIGH Time		15		12		10		ns
4	Instruction Setup Time	(Note 2)	19		17		18		ns
5	Instruction Hold Time	(Note 2)	0		0		0		ns
6	Data Setup Time	(Note 3)	19		17		18		ns
7	Data Hold Time	(Note 3)	0		0		0		ns
8	Control Lines Setup Time	(Note 4)	18		17		16		ns
9	Control Lines Hold Time	(Note 4)	0		0		0		ns
10	F_{31-0} CLOCK-to-Output-Valid F Register Clocked			22		18		15	ns
11	$FLAG_{6-1}$ SIGN CLOCK-to-Output-Valid Status Register Clocked			17		15		13	ns
12	F_{31-0} CLOCK-to-Output-Valid F Register Transparent								
	Flow-Through Mode								
	Multiply-Accumulate			250		215		172	ns
	All Other Operations			200		170		136	ns
	Single-Pipelined Mode								
	Multiply-Accumulate			190		170		136	ns
	All Other Operations			135		110		88	ns
	Double-Pipelined Mode								
	Multiply-Accumulate			135		110		88	ns
13	$FLAG_{6-1}$ SIGN CLOCK-to-Output-Valid Status Register Transparent								
	Flow-Through Mode								
	Multiply-Accumulate			250		215		168	ns
	All Other Operations			195		170		136	ns
	Single-Pipelined Mode								
	Multiply-Accumulate			190		160		128	ns
	All Other Operations			115		100		80	ns
	Double-Pipelined Mode								
	Multiply-Accumulate			115		100		80	ns
14	$\overline{OEF}$, $\overline{OES}$, Disable Time HIGH to Z			18		16		14	ns
15	$\overline{OEF}$, $\overline{OES}$, Disable Time LOW to Z			18		16		14	ns
16	$\overline{OEF}$, $\overline{OES}$, Enable Time Z to HIGH			22		19		18	ns
17	$\overline{OEF}$, $\overline{OES}$, Enable Time Z to LOW			22		19		18	ns
18	FSEL to F_{31-0}			22		18		15	ns
19	MSERR Data-to-Valid Delay			29		27		25	ns

Notes: 1. CLOCK switching characteristics are made relative to 1.5 V.
2. Instruction signals include S/$\overline{DR}$, S/$\overline{DS}$, S/$\overline{DF}$, RM_{2-0}, $PSEL_{3-0}$, $QSEL_{3-0}$, $TSEL_{3-0}$, and I_{13-0}.
3. Data signals include R_{31-0} and S_{31-0}.
4. Control signals include $\overline{ENR}$, $\overline{ENS}$, $\overline{ENF}$, $\overline{ENRF}$, $RFSEL_{2-0}$, and $\overline{ENI}$.

Conditions: A. All inputs/outputs except CLOCK are TTL-compatible for V_{IH}, V_{IL}, and V_{OL}.
B. All outputs are driving 80 pF unless otherwise noted.
C. All setup, hold, and delay times are measured relative to CLOCK at $V_{CC}/2$ volts unless otherwise noted.

SWITCHING CHARACTERISTICS over MILITARY operating range unless otherwise specified

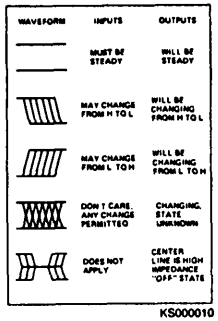
No.	Parameter Description	Test Conditions	Am29C327		Am29C327-1		Am29C327-2		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	
1	CLOCK Period	(Note 1)							
	Flow-Through Mode								
	Multiply-Accumulate		290	DC	245	DC	200	DC	ns
	All Other Operations		230	DC	195	DC	160	DC	ns
	Single-Pipelined Mode								
	Multiply-Accumulate		220	DC	195	DC	160	DC	ns
	All Other Operations		150	DC	120	DC	100	DC	ns
	Double-Pipelined Mode								
	Multiply-Accumulate		150	DC	120	DC	100	DC	ns
2	CLOCK LOW Time		18		15		12		ns
3	CLOCK HIGH Time		18		15		12		ns
4	Instruction Setup Time	(Note 2)	23		21		17		ns
5	Instruction Hold Time	(Note 2)	0		0		0		ns
6	Data Setup Time	(Note 3)	23		21		17		ns
7	Data Hold Time	(Note 3)	0		0		0		ns
8	Control Lines Setup Time	(Note 4)	29		21		17		ns
9	Control Lines Hold Time	(Note 4)	0		0		0		ns
10	F_{31-0} CLOCK-to-Output-Valid F Register Clocked			27		23		19	ns
11	$FLAG_{6-1}$ SIGN CLOCK-to-Output-Valid Status Register Clocked			21		19		16	ns
12	F_{31-0} CLOCK-to-Output-Valid F Register Transparent								
	Flow-Through Mode								
	Multiply-Accumulate			300		260		215	ns
	All Other Operations			240		205		170	ns
	Single-Pipelined Mode								
	Multiply-Accumulate			230		205		170	ns
	All Other Operations			165		135		110	ns
	Double-Pipelined Mode								
	Multiply-Accumulate			165		135		110	ns
13	$FLAG_{6-1}$ SIGN CLOCK-to-Output-Valid Status Register Transparent								
	Flow-Through Mode								
	Multiply-Accumulate			300		255		215	ns
	All Other Operations			235		205		170	ns
	Single-Pipelined Mode								
	Multiply-Accumulate			230		195		160	ns
	All Other Operations			140		125		100	ns
	Double-Pipelined Mode								
	Multiply-Accumulate			140		125		100	ns
14	$\overline{OEF}$, $\overline{OES}$, Disable Time HIGH to Z			22		20		18	ns
15	$\overline{OEF}$, $\overline{OES}$, Disable Time LOW to Z			22		20		16	ns
16	$\overline{OEF}$, $\overline{OES}$, Enable Time Z to HIGH			27		23		19	ns
17	$\overline{OEF}$, $\overline{OES}$, Enable Time Z to LOW			27		23		19	ns
18	FSEL to F_{31-0}			27		22		18	ns
19	MSERR Data-to-Valid Delay			34		33		27	ns

Notes: 1. CLOCK switching characteristics are made relative to 1.5 V.
2. Instruction signals include S/$\overline{DR}$, S/$\overline{DS}$, S/$\overline{DF}$, RM_{2-0}, $PSEL_{3-0}$, $QSEL_{3-0}$, $TSEL_{3-0}$, and I_{13-0}.
3. Data signals include R_{31-0} and S_{31-0}.
4. Control signals include $\overline{ENR}$, $\overline{ENS}$, $\overline{ENF}$, $\overline{ENRF}$, $RFSEL_{2-0}$, and $\overline{ENI}$.

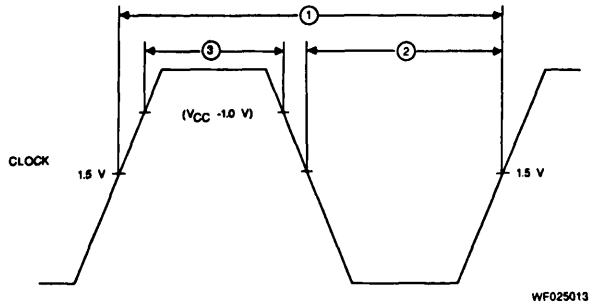
Conditions: A. All inputs/outputs except CLOCK are TTL-compatible for V_{IH}, V_{IL}, and V_{OL}.
B. All outputs are driving 80 pF unless otherwise noted.
C. All setup, hold, and delay times are measured relative to CLOCK at $V_{CC}/2$ volts unless otherwise noted.

SWITCHING WAVEFORMS

KEY TO SWITCHING WAVEFORMS



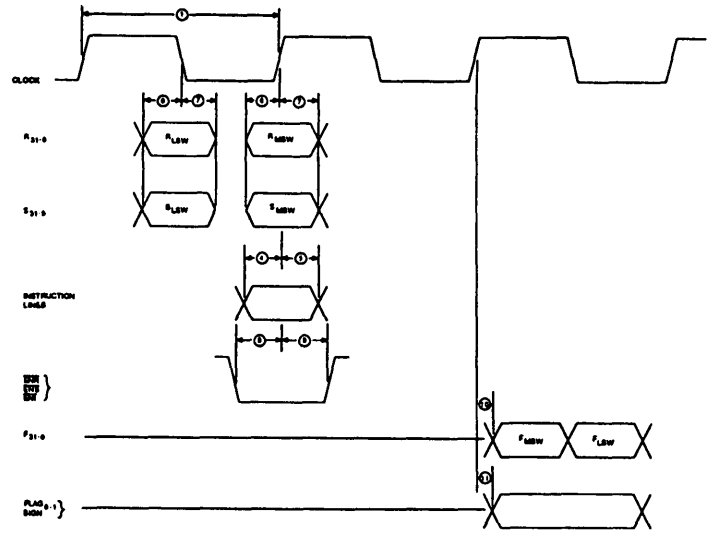
KS000010



WF025013

Input Clock Timing

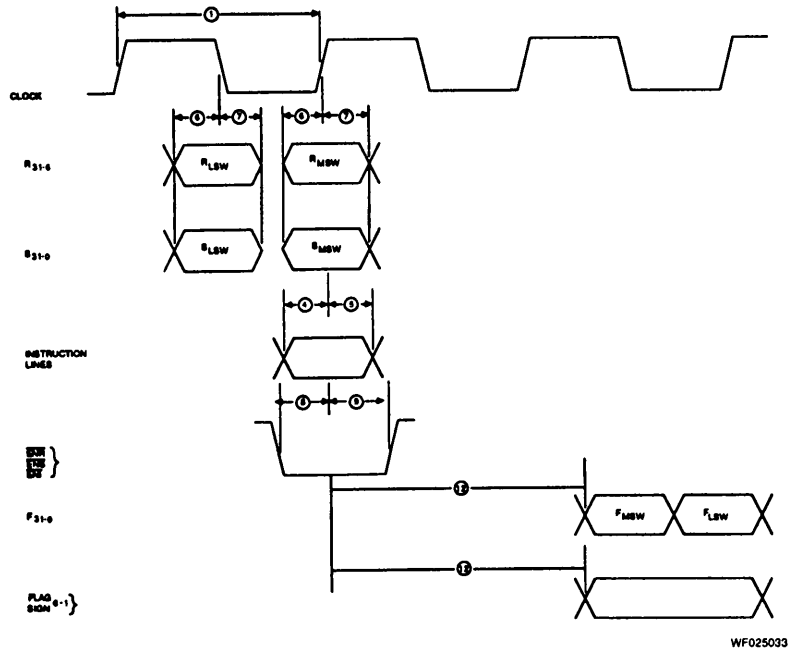
SWITCHING WAVEFORMS (Cont'd.)



WF025023

Timing of Operations with F Register and Status Clocked. Assumes 32-Bit Bus, Single-Cycle, LSW-First Input Mode and Flow-Through Operation

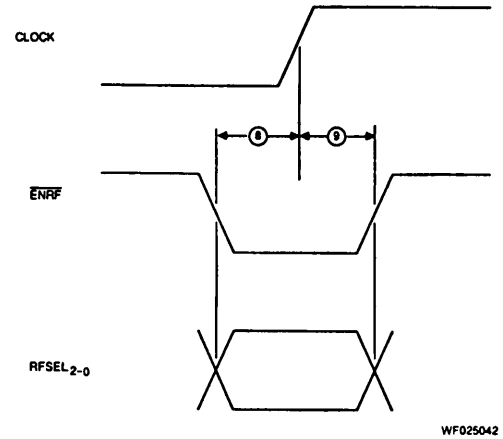
SWITCHING WAVEFORMS (Cont'd.)



WF025033

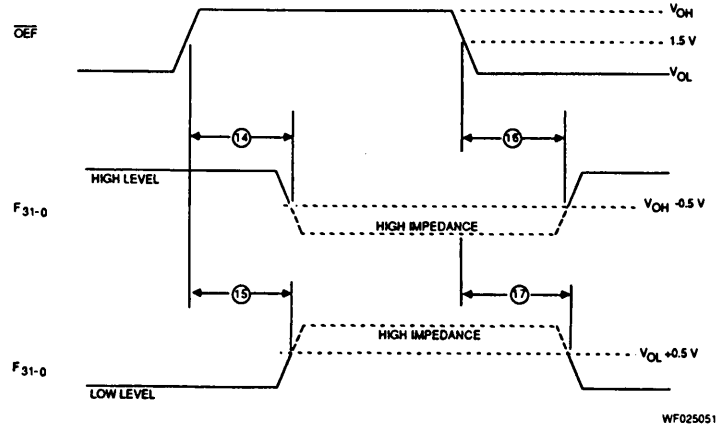
Timing of Operations with F-Register and Status Register in Feedthrough Mode. Assumes 32-Bit Bus, Single-Cycle, LSW-First Input Mode and Flow-Through Operation.

SWITCHING WAVEFORMS (Cont'd.)



WF025042

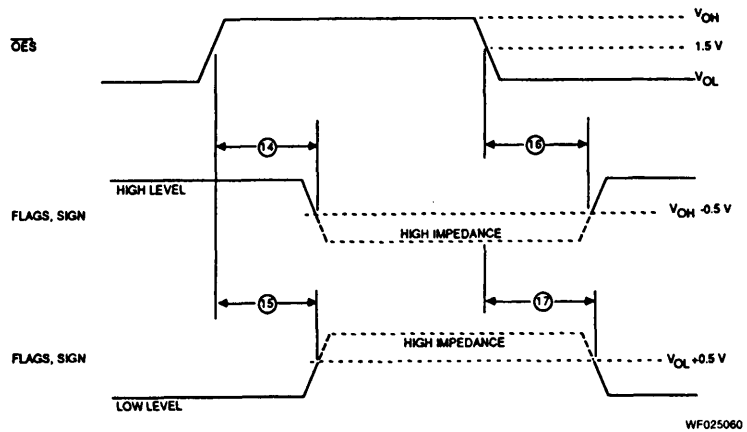
Register File Control Timing



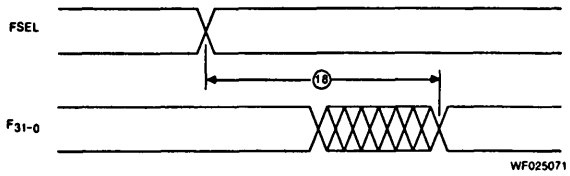
WF025051

Enable/Disable Timing for F_{0-31}

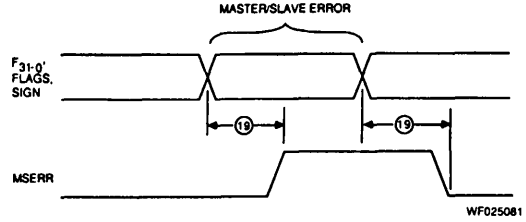
SV ITCHING WAVEFORMS (Cont'd.)



Enable/Disable Timing for $FLAG_{1-6}$ and SIGN

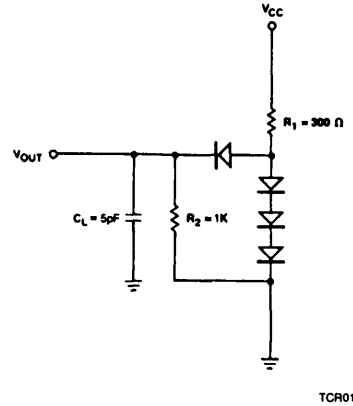


Output Selection Timing

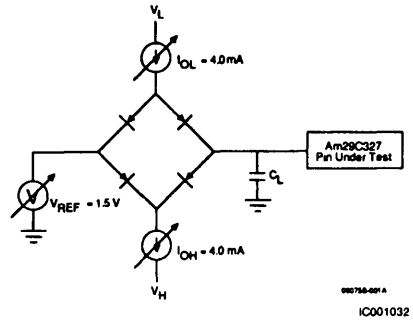


Master/Slave Timing (Assumes SLAVE Mode)

SWITCHING TEST CIRCUITS

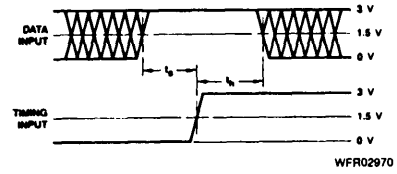


A. Three-State Outputs



B. All Other Outputs

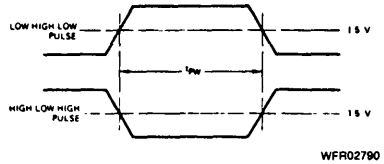
SWITCHING TEST WAVEFORMS



WFR02970

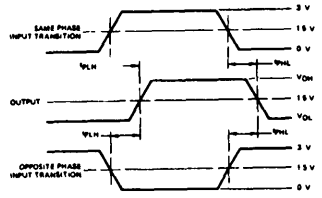
Notes: 1. Diagram shown for HIGH data only. Output transition may be opposite sense.
2. Cross-hatched area is don't care condition.

Setup, Hold, and Release Times



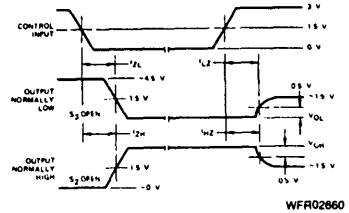
WFR02790

Pulse Width



WFR02980

Propagation Delay



WFR02660

Notes: 1. Diagram shown for Input Control Enable-LOW and Input Control Disable-HIGH.
2. S_1, S_2 and S_3 of Load Circuit are closed except where shown.

Enable and Disable Times

TEST PHILOSOPHY AND METHODS

The following eight points describe AMD's philosophy for high volume, high speed automatic testing.

1. Ensure that the part is adequately decoupled at the test head. Large changes in V_{CC} current as the device switches may cause erroneous function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may start to oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an output transition, ground current may change by as much as 400 mA in 5-8 ns. Inductance in the ground cable may allow the ground pin at the device to rise by hundreds of millivolts momentarily.
4. Use extreme care in defining point input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leq 0$ V and $V_{IH} \geq 3.0$ V for AC tests.
5. To simplify failure analysis, programs should be designed to perform DC, Function, and AC tests as three distinct groups of tests.
6. Capacitive Loading for AC Testing

   Automatic testers and their associated hardware have stray capacitance that varies from one type of tester to another, but is generally around 50 pF. This, of course, makes it impossible to make direct measurements of parameters which call for smaller capacitive load than the associated stray capacitance. Typical examples of this are the so-called "float delays," which measure the propagation delays into the high-impedance state and are usually specified at a load capacitance of 5.0 pF. In these cases, the test is performed at the higher load capacitance (typically 50 pF), and engineering correlations based on data taken with a bench setup are used to predict the result at the lower capacitance.

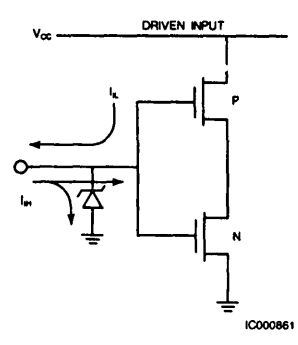
   Similarly, a product may be specified at more than one capacitive load. Since the typical automatic tester is not capable of switching loads in mid-test, it is impossible to make measurements at both capacitances even though they may both be greater than the stray capacitance. In these cases, a measurement is made at one of the two capacitances. The result at the other capacitance is predicted from engineering correlations based on data taken with a bench setup and the knowledge that certain DC measurements (I_{OH}, I_{OL} for example) have already been taken and are within spec. In some cases, special DC tests are performed in order to facilitate this correlation.
7. Threshold Testing

   The noise associated with automatic testing (due to the long, inductive cables) and the high gain of the tested device when in the vicinity of the actual device threshold, frequently give rise to oscillations when testing high speed circuits. These oscillations are not indicative of a reject device, but instead of an overtaxed test system. To minimize this problem, thresholds are tested at least once for each input pin. Thereafter, "hard" high and low levels are used for other tests. Generally this means that function and AC testing are performed at "hard" input levels rather than at V_{IL} Max. and V_{IH} Min.
8. AC Testing

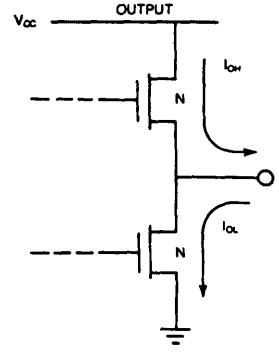
   Occasionally, parameters are specified that cannot be measured directly on automatic testers because of tester limitations. Data input hold times often fall into this category. In these cases, the parameter in question is guaranteed by correlating these tests with other AC tests that have been performed. These correlations are arrived at by the cognizant engineer by using precise bench measurements in conjunction with the knowledge that certain DC parameters have already been measured and are within spec.

   In some cases, certain AC tests are redundant, since they can be shown to be predicted by some other tests which have already been performed. In these cases, the redundant tests are not performed.

INPUT/OUTPUT CIRCUIT DIAGRAMS



IC000861

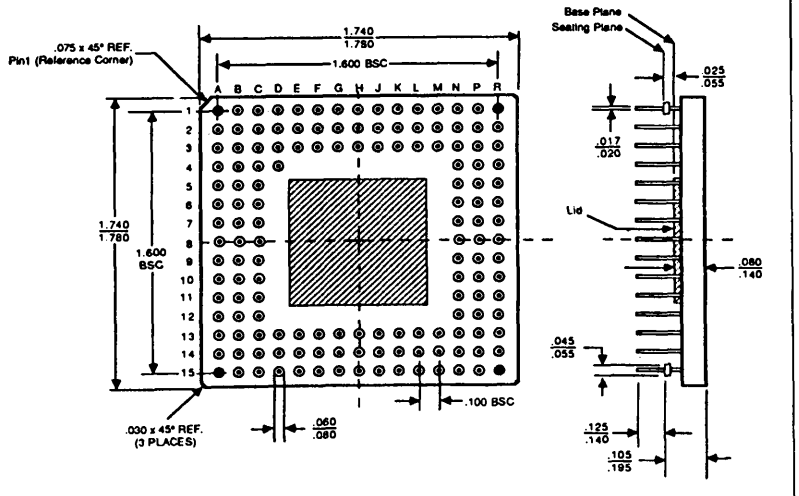


IC000870

PHYSICAL DIMENSIONS

CGX169

BOTTOM VIEW
(Pins facing up)



PID# 07322C

*For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

APPENDICES

APPENDIX A — DATA FORMATS

The following data formats are supported: 32-bit integer, 64-bit integer, IEEE single-precision, IEEE double-precision, DEC F, DEC D, DEC G, IBM single-precision, and IBM double-precision.

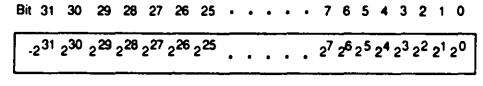
The primary and alternate floating-point formats are selected by mode register bits M3 to M0. The user may select between floating-point operations and integer operations by means of instruction bit I5.

The nine supported formats are described below:

Integer Formats

32-Bit Integer

The 32-bit integer word is arranged as follows:

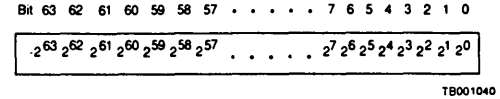


TB001030

The 32-bit word is interpreted as a two's-complement integer. For integer multiplications, the user has the option of interpreting integers as unsigned. An unsigned single-precision integer has a format similar to that of the two's-complement integer, but with an MSB weight of 2^{31}.

64-Bit Integer

The 64-bit integer word is arranged as follows:



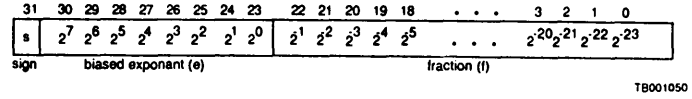
TB001040

The 64-bit word is interpreted as a two's-complement integer. For integer multiplications, the user has the option of interpreting integers as unsigned. An unsigned double-precision integer has a format similar to that of the two's-complement integer, but with an MSB weight of 2^{63}.

IEEE Formats

IEEE Single-Precision

The IEEE single-precision word is 32 bits wide and is arranged in the format as follows:



TB001050

The floating-point word is divided into three fields: a single-bit sign, an 8-bit biased exponent, and a 23-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers. Zero may have either sign.

The biased exponent is an 8-bit unsigned integer representing a multiplicative factor of some power of two. The bias value is 127. If, for example, the multiplicative value for a floating-point number is to be 2^{a}, the value of the biased exponent is a + 127, where "a" is the true exponent.

The fraction is a 23-bit unsigned fractional field containing the 23 least-significant bits of the floating-point number's 24-bit mantissa. The weight of the fraction's most-significant bit is 2^{-1}. The weight of the least-significant bit is 2^{-23}.

An IEEE floating-point number is evaluated or interpreted as follows:

If e = 255 and f ≠ 0	value = NaN	Not-a-Number
If e = 255 and f = 0	value = $(-1)^{s}\infty$	Infinity
If 0 < e < 255	value = $(-1)^{s}2^{e-127}(1.f)$	Normalized number
If e = 0 and f ≠ 0	value = $(-1)^{s}2^{-126}(0.f)$	Denormalized number
If a = 0 and f = 0	value = $(-1)^{s}0$	Zero

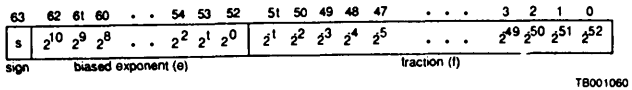
Infinity: Infinity can have either a positive or negative sign. The interpretetion of infinities is determined by the Atfine/Projective select input AFF/$\overline{\text{PROJ}}$.

NaN: A NaN is interpreted as e signal or symbol. NaNs are used to indicete invelid operetions, end es e means of passing process status through e series of celculetions. They erise in two ways: either genereted by the Am29C327 to indicete an invelid operetion, or provided by the user as en input. A signaling NeN has the MSB of its fraction set to 0 and et least one of the remeining fraction bits set to 1. A quiet NeN has the MSB of its fraction set to 1.

The IEEE formet is fully described in ANSI/IEEE Stendard 754-1985.

IEEE Double-Precision

The IEEE double-precision word is 64 bits wide end is errenged in the formet shown below:



TB001060

The floating-point word is divided into three fields: e single-bit sign, en 11-bit biased exponent, and a 52-bit frection.

The sign bit is 0 for positive numbers end 1 for negetive numbers; zero mey have either sign.

The biased exponent is an 11-bit unsigned integer represent-ing e multiplicetive fector of some power of two. The bies value is 1023. If, for exemple, the multiplicative value for e floating-point number is to be 2^a, the velue of the biased exponent is e + 1023, where "a" is the true exponent.

The frection is e 52-bit unsigned frectionel field containing the 52 leest-significant bits of the floating-point number's 53-bit mentissa. The weight of the fraction's most-significent bit is 2^{-1}. The weight of the least-significent bit is 2^{-52}.

An IEEE floeting-point number is evelueted or interpreted es follows:

If e = 2047 end f ≠ 0 ..... value = Reserved operend Not-e-Number
If e = 2047 end f = 0 ..... velue = $(-1)^s\infty$ Infinity
If 0 < e < 2047 ..... velue = $(-1)^s 2^{e-1023}(1.f)$ Normalized number
If e = 0 end f ≠ 0 ..... velue = $(-1)^s 2^{-1022}(0.f)$ Denormelized number
If e = 0 end f = 0 ..... velue = $(-1)^s 0$ Zero

Infinity: Infinity cen heve either e positive or negative sign. The interpretetion of infinities is determined by the Atfine/Projective select input AFF/$\overline{\text{PROJ}}$.

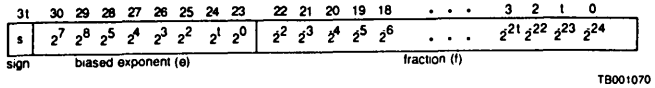
NaN: A NaN is interpreted es e signel or symbol. NeNs are used to indicate invalid operations, end es a means of passing process stetus through a series of celculations. They erise in two weys: either genereted by the Am29C327 to indicete an invelid operetion, or provided by the user es en input. A signeling NaN has the MSB of its frection set to 0 end at least one of the remaining fraction bits set to 1. A quiet NaN hes the MSB of its frection set to 1.

The IEEE formet is fully described in ANSI/IEEE Stendard 754-1985.

DEC Formats

DEC F

The DEC F word is 32 bits wide end is errenged in the format shown below:



TB001070

The floating-point word is divided into three fields: a single-bit sign, en 8-bit biesed exponent, end e 23-bit frection.

The sign bit is 0 for positive numbers and 1 for negetive numbers; zero hes e positive sign.

The biesed exponent is an 8-bit unsigned integer representing e multiplicative factor of some power of two. The bies value is 128. If, for example, the multiplicetive value for e floeting-point number is to be 2^a, the velue of the biased exponent is e + 128, where "a" is the true exponent.

The fraction is e 23-bit unsigned frectional field conteining the 23 least-significant bits of the floeting-point number's 24-bit mantissa. The weight of the frection's most-significant bit is 2^{-2}. The weight of the least-significent bit is 2^{-24}.

A DEC F floating-point number is evelueted or interpreted as follows:

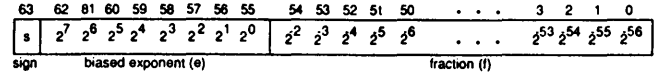
If e ≠ 0 ..... value = $(-1)^s 2^{e-128}(0.1f)$
If s = 0 and e = 0 ..... value = 0
If s = 1 and e = 0 ..... value = DEC-Reserved Operand

DEC-Reserved Operand: A DEC-Reserved Operend is interpreted es e signal or symbol. DEC-Reserved Operands are used to indicate invelid operetions and operetions whose results heve overflowed the destinetion formet. They may elso be used to pass symbolic information from one celculation to enother.

The DEC formets ere fully described in the VAX Architecture Menuel.

DEC D

The DEC D word is 64 bits wide end is arranged in the format shown below:



TB001080

The floeting-point word is divided into three fields: e single-bit sign, en 8-bit biesed exponent, and a 55-bit fraction.

The sign bit is 0 for positive numbers end 1 for negative numbers; zero has e positive sign.

The biased exponent is en 8-bit unsigned integer representing a multiplicetive fector of some power of two. The bias value is 128. If, for exemple, the multiplicative value for a floating-point number is to be 2^a, the value of the biased exponent is e + 128, where "a" is the true exponent.

The frection is e 55-bit unsigned fractionel field containing the 55 least-significent bits of the floeting-point number's 56-bit mentissa. The weight of the frection's most-significant bit is 2^{-2}. The weight of the least-significant bit is 2^{-56}.

A DEC D floeting-point number is eveluated or interpreted es follows:

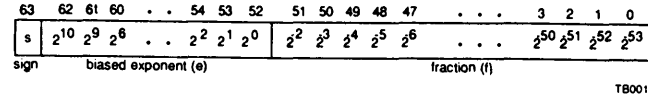
If e ≠ 0 ..... value = $(-1)^s 2^{e-128}(0.1f)$
If s = 0 end e = 0 ..... velue = 0
If s = 1 end e = 0 ..... velue = DEC-Reserved Operand

DEC-Reserved Operand: A DEC-Reserved Operend is interpreted as e signal or symbol. DEC-Reserved Operands are used to indicate invalid operations end operations whose results heve overflowed the destinetion formet. They may elso be used to pass symbolic information from one calculation to enother.

The DEC formets ere fully described in the VAX Architecture Menual.

DEC G

The DEC G word is 64 bits wide end is errenged in the formet shown below:



TB001090

The floating-point word is divided into three fields: e single-bit sign, an 11-bit biesed exponent, and a 52-bit fraction.

The sign bit is 0 for positive numbers end 1 for negative numbers; zero hes e positive sign.

The biesed exponent is en 11-bit unsigned integer represent-ing a multiplicative fector of some power of two. The bias velue is 1024. If, for exemple, the multiplicetive value for e floating-point number is to be 2^a, the velue of the biased exponent is e + 1024, where "e" is the true exponent.

The fraction is e 52-bit unsigned frectionel field containing the 52 least-significant bits of the floating-point number's 53-bit mentisse. The weight of the fraction's most-significant bit is 2^{-2}. The weight of the least-significant bit is 2^{-53}.

A DEC G floating-point number is evalueted or interpreted es follows:

If e ≠ 0 ..... velue = $(-1)^s 2^{e-1024}(0.1f)$
If s = 0 and e = 0 ..... value = 0
If s = 1 and e = 0 ..... value = DEC-Reserved Operand

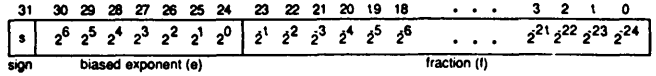
DEC-Reserved Operand: A DEC-Reserved Operand is interpreted as e signel or symbol. DEC-Reserved Operands are used to indicate invalid operetions end operations whose results heve overflowed the destinetion format. They may elso be used to pess symbolic intormetion from one calculation to enother.

The DEC formats are fully descnbed in the VAX Architecture Manual.

IBM Formats

IBM Single-Precision

The IBM singla-precision word is 32 bits wide end is erranged in the format shown below:



TB001100

The floating-point word is divided into three fields: a single-bit sign, e 7-bit biesed exponent, end e 24-bit fraction.

The sign bit is 0 for positive numbers and t for negetive numbers; e True-zero hes a positive sign.

The biased exponent is a 7-bit unsigned integer raprasenting a multiplicetiva fector of some power of 16. Tha bies value is 64. If, for example, the multiplicetiva velua for e floating-point number is to be 16^a, the value of the biased exponent is a + 64, where "a" is the true exponent.

The fraction is a 24-bit unsigned fractionel field containing the 24 least-significant bits of the floating-point number's 25-bit mantissa. The weight of the frection's most-significant bit is 2^{-1}. The weight of the least-significent bit is 2^{-24}.

An IBM floating-point number is avaluated or intarpreted as follows:

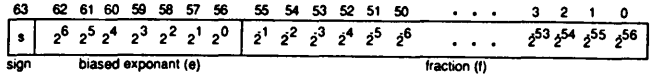
$$value = (-1)^s 16^{e-64}(0.f)$$

Zero: There ara two classas of zero. If the sign, biased exponant and fraction are all zero, the operand is known as a "True Zaro." If tha fraction is zaro, but the sign and biased exponent are not both zero, the operand is known es e "Floating-point Zero."

Tha IBM formet is fully described in the IBM System/370 Principles of Operation Manuel.

IBM Double-Precision

The IBM double-precision word is 64 bits wide end is arranged in the format shown below:



TB001110

The floating-point word is divided into three fields: a singla-bit sign, e 7-bit biased exponent, end a 56-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers; a True-zaro hes a positive sign.

The biesed exponent is a 7-bit unsigned integer representing e multiplicetive factor of some power of 16. The bies value is 64. If, for exampla, the multiplicative value for e floating-point number is to be 16^a, tha value of the biased exponent ia a + 64, where "e" is the trua exponent.

The frection is a 56-bit unsigned fractionel field conteining the 56 laast-significent bits of tha floating-point number's 57-bit mentissa. The weight of the fraction's most-significent bit is 2^{-1}. The weight of the laest-significent bit is 2^{-56}.

An IBM floating-point number is eveluated or interprated as follows:

$$value = (-1)^s 16^{e-64}(0.f)$$

Zero: There are two classes of zero. If the sign, biased exponent end frection ara ell zero, the operand is known as a "True Zero." If the fraction is zero, but the sign and biesed exponent are not both zero, tha operend ia known es a "Floeting-point Zero."

The IBM formet is fully described in the IBM System/370 Principlas of Operation Menuel.

APPENDIX B — ROUNDING MODES

Tha Am29C327 provides six rounding modes for floating-point operations, and for integer multiplication. The rounding mode for an operetion is selected by the input pins RM_2-RM_0.

RM_2	RM_1	RM_0	Round Mode
0	0	0	Round to Neerest (IEEE)
0	0	1	Round to Minus Infinity
0	t	0	Round to Plus Infinity
0	t	1	Round to Zero
t	0	0	Round to Neerest (DEC)
1	0	t	Round Away From Zero
t	1	X	Illegal Value

Round to Nearest (IEEE)

Tha infinitaly pracise result of an operation is rounded to the closast reprasantabla valua in tha destination format. If the infinitely precisa rasult is exactly halfway betwaan two rapresantations, it is roundad to tha rapresentation heving a laast-significant bit of zaro.

Round to Minus Infinity (IEEE)

The infinitaly precise result of an operation is rounded to the closest reprasentabla value in the destinetion format that is lass than or equal to the infinitaly precise result. This rounding mode conforms to the "round to minus infinity" mode described in the IEEE Floating-Point Standard.

Tha IEEE standard spacifies that all four "IEEE" modes be available so that tha usar may salact the moda most appropriata for tha algorithm being axecutad. The DEC standard spacifias that two rounding modas be available - Round-to-Naarast (DEC) and Round-to-Zero. Tha IBM stendard specifias that all operations be performed using the Round-to-Zero mode.

It should be noted, however, that the Am29C327 permits any of the supported rounding modes to be selected, regardless of the format of the operation. It is permissabla to use ona of the IEEE rounding modes with an IBM operation, or DEC rounding with an IEEE operetion, or any other possible combination. For those integer operations where rounding is performed, any rounding mode may be chosen. This flexibility ellows the user to select the mode most appropriate for the erithmetic environment in which the processor is operating.

Round to Plus Infinity (IEEE)

The infinitaly precise result of an operation is rounded to the closest reprasentable value in tha destination format thet is greater then or equal to the infinitely precise result.

Round to Zero (IEEE)

The infinitely precise result of en operation is rounded to the closast raprasentabla value in the destination format whose magnitude is less than or equal to the infinitaly precise result.

Round to Nearest (DEC)

The infinitaly precisa rasult of en operation is rounded to the closest raprasentabla valua in the dastination format. If tha infinitaly precise rasult is exactly halfway between two represantations, it is rounded to the reprasantation having the graeter magnitude.

Round Away from Zero

The infinitely precise rasult of an operation is rounded to the closast raprasentable value in tha dastination format whose magnitude is greater than or equal to the infinitely precise rasult.

A graphical rapresantation of these rounding modas is shown in Figures B1-1 end B1-2.

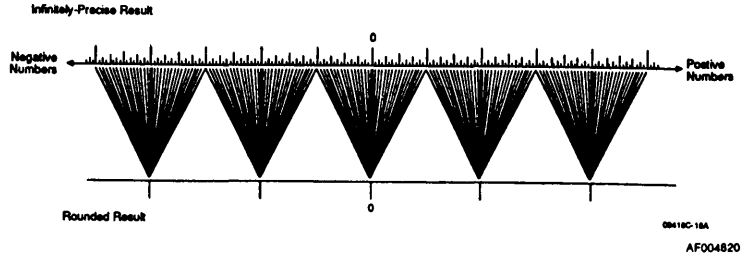


Figure B1. Illustration of IEEE Round-to-Nearest Rounding Mode

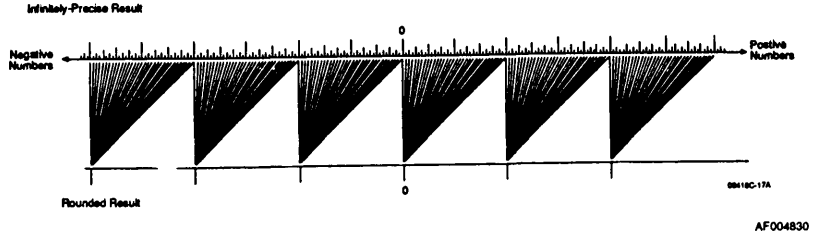


Figure B2. Illustration of IEEE Round-to-Minus-Infinity Rounding Mode

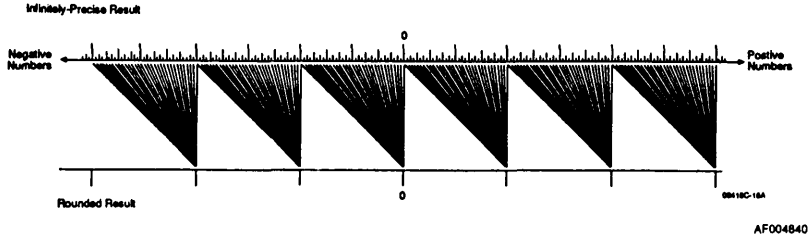


Figure B3. Illustration of IEEE Round-to-Plus-Infinity Rounding Mode

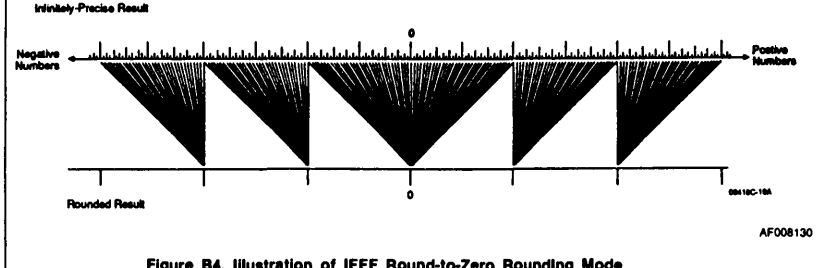


Figure B4. Illustration of IEEE Round-to-Zero Rounding Mode

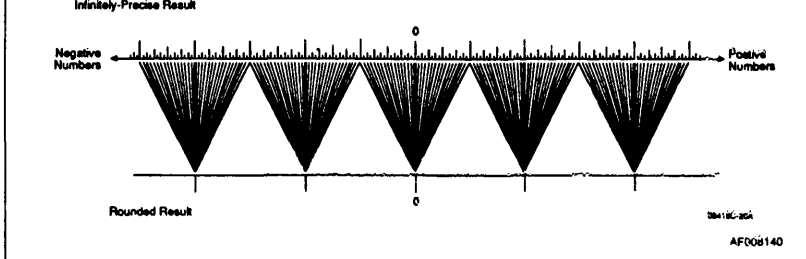


Figure B5. Illustration of DEC Round-to-Nearest Rounding Mode

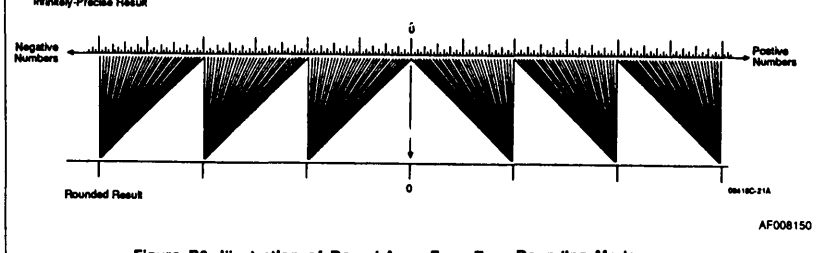


Figure B6. Illustration of Round-Away-From-Zero Rounding Mode

APPENDIX C — DEVIATIONS FROM FLOATING-POINT STANDARDS

There are several cases in which the implementation of the IEEE, DEC and IBM floating-point standards in the Am29C327 differs from the formal definitions of those standards. This appendix describes these deviations from the standards.

Deviations from the IEEE Standard

Section 7.3 of the IEEE-754 standard specifies that "Trapped overflow on conversion from a binary floating-point format shall deliver to the trap handler a result in that or a wider format, possibly with the exponent bias adjusted, but rounded to the destination's precision."

According to the IEEE standard, then, if a double-to-single IEEE operation overflows while traps are enabled, the result is a double-precision operand, rounded to single-precision width (23-bit fraction), together with a correctly-adjusted (double-precision) exponent and the appropriate flags for a trapped overflow.

In the case of an overflow in any IEEE operation, the Am29C327 returns a result in the destination format specified by the user, rounded to that destination format.

In the case of the double-to-single overflow described above, the result from the Am29C327 is a single-precision operand, together with a correctly-adjusted (single-precision) exponent and the appropriate flags for a trapped overflow.

A simple example serves to illustrate the discrepancy, by describing the conversion of the double-precision IEEE number 52B12345676 9ABCD to single-precision, with traps enabled and the round-to-nearest rounding mode selected. This number is too large to be rapresented in single-precision format.

According to the IEEE standard, the result of this operation is the double-precision number 52B1234560000000, comprising the double-precision exponent of the input and a fraction truncated to 23 bits, together with flags V and X.

When the operation ia performed in the Am29C327, however, using the " F' = P' " operation with appropriate precision controls, the result is the single-precision number 75891A2B, comprising the single-precision (overflowed) exponent reduced by 192 (decimal) and a single-precision fraction, together with flags V and X.

It should be noted that trapped operation is an optional part of the IEEE standard. Full adherence to the IEEE specification of trapped operation is therefore not necessary to ensure compliance with IEEE-754.

Deviations from the DEC F, DEC D and DEC G Standards

The DEC F, DEC D and DEC G standards, as implemented in the Am29C327, differ from the implementations in a VAX only in the way in which the sub-fields of the floating-point word are arranged. The differences are listed in Table C1.

TABLE C1. DIFFERENCES IN Am29C327 AND DEC FLOATING-POINT FORMATS

	Am29C327 Arrangement	VAX Arrangement
DEC F	sign: bit 31 exponent: bits 30-23 fraction: bits 22-0	sign: bit 15 exponent: bits 14-7 fraction: bits 6-0, bits 31-16
DEC D	sign: bit 63 exponent: bits 62-55 fraction: bits 54-0	sign: bit 15 exponent: bits 14-7 fraction: bits 6-0, bits 31-16, bits 47-32, bits 63-46
DEC G	sign: bit 63 exponent: bits 62-52 fraction: bits 51-0	sign: bit 15 exponent: bits 14-4 fraction: bits 3-0, bits 31-16, bits 47-32, bits 63-48

The discrepancies are shown graphically in Figure C1. Within each exponent and fraction field, the shading illustrates the weighting of the bits, from the MSB to LSB.

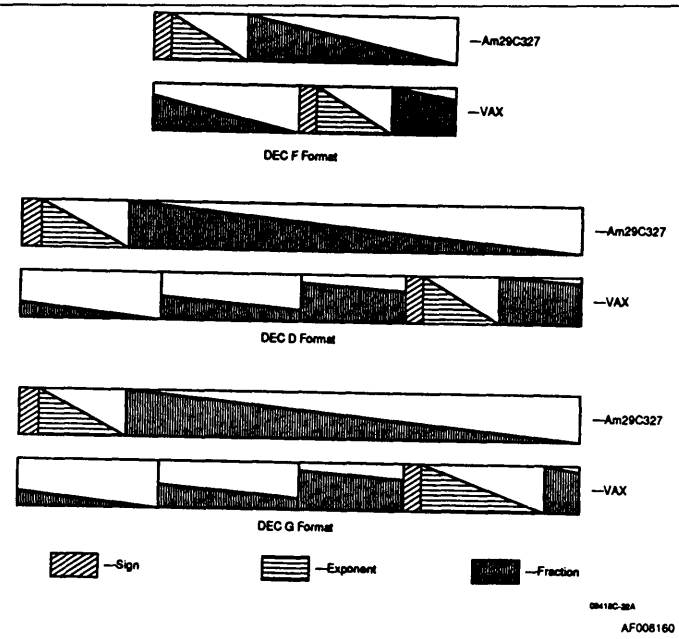


Figure C1. Differences in DEC Representations Between the Am29C327 and the VAX

Deviations from the IBM Standard

The Am29C327's deviations from the IBM standard may be summarized as follows, assuming that the user has selected the round-to-nearest rounding mode:

1. The Am29C327 provides more guard bits in its internal format than specified by the IBM standard. With certain combinations of input operands, the Am29C327 produces more accurate results than a standard IBM processor for instructions based on addition operations and comparisons (i.e. IBM instructions with an op-code, I4-0, of 00001 or 00011).
2. The discrepancies are much larger for single-precision operations than double-precision operations, because the difference in the number of guard bits is much greater (33 more for single, one more for double).
3. There is no universal rule for determining whether a given set of input operands will result in a discrepancy. Provided the conditions in (1) above are met, the user must examine each operation on a case-by-case basis, taking into account the input operands and the internal formats discussed in this section.
4. The Am29C327 does not produce unnormalized results from additions. The results of all addition operations are renormalized.

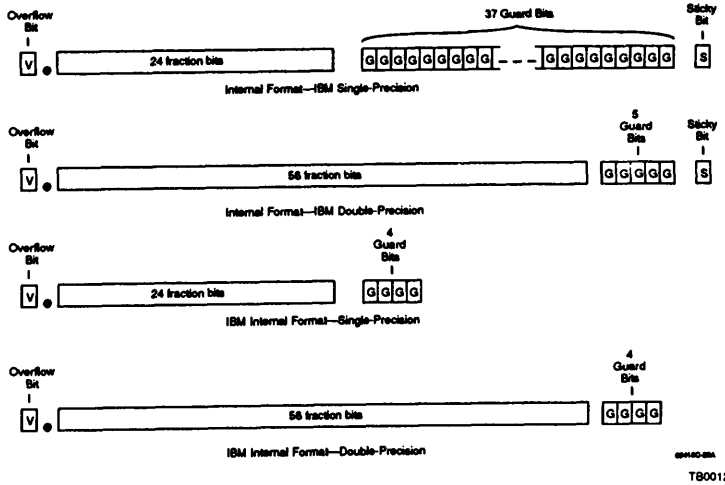


Figure C2. Differences in Internal Mantissa Formats of an IBM CPU and the Am29C327

It should be mentioned that the discrepancies due to the above effects, for both additions and multiplications, are typically insignificant when compared to the magnitude of the result itself. They become an important issue only when the user requires exact compatibility with an existing implementation of the IBM standard.

The term "accuracy" in this discussion is used to describe the difference between the final result of an operation and its infinitely-precise result. In the context of adherence to a standard, however, "accuracy" might better be defined as the difference between the final result of an operation executed in the Am29C327 and the final result of the same operation executed in a CPU which exactly meets that standard.