VLBA Technical Report No. 45
THE VLBA CORRELATOR
MAC and LTA SUB-SYSTEMS
VOLUME 1 of 2

Chuck Broadwell

Ray Escoffier

January 1, 1999

macltaman.vol1.doc Microsoft Word not under SCCS control

master file stored in /corrdwgs/manuals/macltaman/SCCS
(where the SCCS file would have been stored)

This page intentionally left blank.

TABLE OF CONTENTS

VOLUME 1

TABLE OF FIGURES

VOLUME 1

1 The VLBA Correlator MAC and LTA Sub-Systems

1.1 Introduction

The MAC and LTA sub-systems consist of the following card types:

1) Master Control Card (MCC)
2) Multiply and Accumulate Card (MAC)
3) Long Term Accumulator Card (LTA)

The MCC provides master control signals to the complete correlator system, as well as controlling the MAC cards.

The MAC cards come after the FFT cards in the data flow path of the VLBA Correlator System (see VLBA Technical Report Number 44). The spectral results produced by the FFT process, for each antenna, are cross multiplied with results from all other antennas and the results are accumulated on a short term basis in the MAC cards. The short term accumulation cycle is 131 msec.

Results from the MAC cards are transferred to the Long Term Accumulator (LTA) every 131 msec and usually accumulated for longer intervals. The LTA accumulates for integer multiples of the 131 msec interval. Results from the LTA are transferred to the Digital Filter (FIR), covered in VLBA Technical Report Number 46.

The MCC and MAC cards are located in the top card bin in each of the correlator racks. There is one MCC card per rack, and 15 MAC cards per rack. The LTA card is located in the FFT bin of rack 3 only. See Figure 1 for the location of these bins. There is only one LTA card in the system.

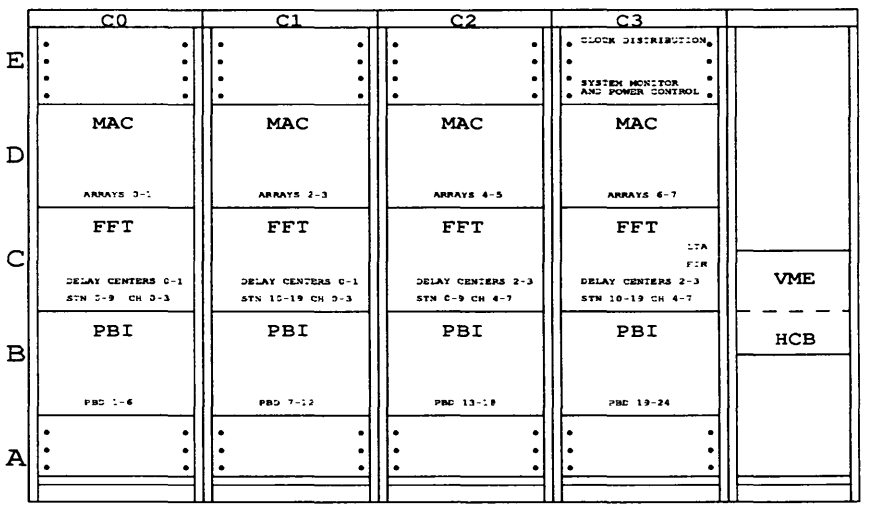


Figure 1 VLBA Correlator Rack Layout

Figure 2 shows the card layout in a MAC bin.

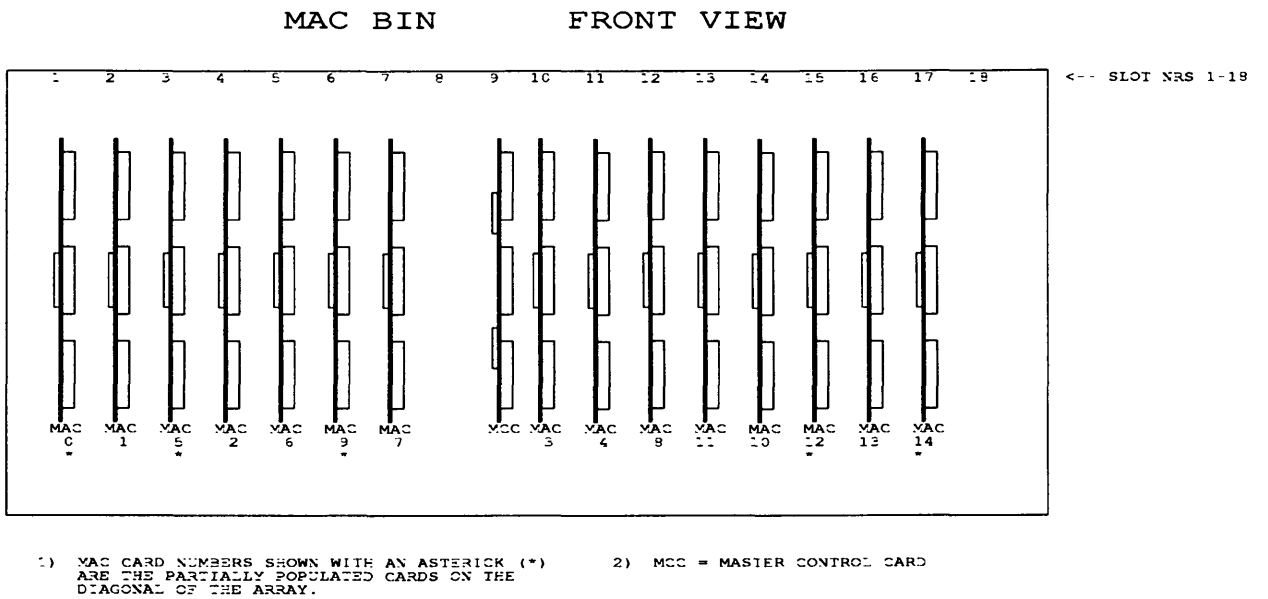


Figure 2 MAC Bin Card Layout

1.2 VLBA Correlator Drawings

All hardware and firmware related files for the VLBA correlator are maintained under SCCS version control, in one of two areas. These two areas are identified as the vlbsoft and corrdwgs areas. The path to vlbsoft is /home/magnolia2/vlbsoft and the path to corrdwgs is /home/azalea/corrdwgs (as of May 1998). All drawing filenames are lower case, but may appear in documentation as either upper or lower case.

The vlbsoft area is where all source code for the correlator is maintained. It is also used to store all source files that are processed in any manner to produce objects that are downloaded into the correlator, or programmed into devices that are installed in the correlator.

The corrdwgs area is where Orcad schematic files, netlists, partlists, etc. are maintained.

Section 5 of this Technical Report contains a complete list of all drawings for the MCC, MAC and LTA cards.

Volume 2 of this Technical Report contains many of the schematics referred to in Volume 1. Drawings for test fixtures are not included in this report.

Copies of many Postscript files for drawings are presently stored in the /home/w5uxh/plots directory.

1.3 Test Fixtures

To support testing of the MCC, MAC and LTA cards, there are several test fixtures available. The PBI test fixture, described in Technical Report No. 43, supports the MCC card. The FFT test fixture, described in Technical Report No. 44, supports the MAC cards. A separate test fixture, the LTA-FIR test fixture, supports the LTA and FIR (Digital Filter) cards. This test fixture has no built in test features. It consists of slots for the LTA and FIR cards and a HCB interface to the VME system.

The VLBA Correlator Testbed Rack provides additional test capability for all system cards.

This page intentionally left blank.

2 The Master Control Card (MCC)

2.1 Introduction

The main function of the master control card (MCC) is to provide timing and control signals throughout the VLBA correlator. A summary of the specific functions provided by the MCC card is given below:

1) Provide FFT cycle, fringe cycle, and integration cycle sequencers that can produce timing signals required by other sections of the correlator.

2) Provide control signals for the MAC cards in the cross-multiplier bin.

3) Duplicate, for validity signals, the 4-into-1 data muxes that exist on the FFT cards.

4) Provide tape validity integrators for 2-channels (all 20-stations) and pulsar validity integrators for the pulsar gate generators of one FCC.

5) Provide differential drivers for the 18-bit short term integrator bin output bus into the LTA.

6) Provide a data tap for the short term integrator output for testing purposes.

7) Provide correctly timed validity gate signals to the FFT cards.

8) Provide an HCB (Hardware Control Bus) communication link with the real time computer system (RTS).

9) Load the MAC ASIC control words.

10) Provide a test PBD frame for system testing.

There are 4 MCC cards in the system, one in each MAC bin. The MCC card in rack 3 is a master MCC and its sequencers free-run on the 32-MHz system clock. The MCC cards in racks 0, 1, and 2 are slaved to the rack 3 card. See Figure 3 for a block diagram of the MCC.

2.2 Circuit Description

Refer to the MCC schematic sheets, L027D01.SCH - L027D04.SCH in Volume 2, for the following circuit descriptions.

2.2.1 The FFT Cycle Sequencer

The FFT cycle sequencer of the MCC is seen on sheet 2 of the MCC schematic in the upper left hand corner. This logic provides a 516-state sequencer which is synchronized, in the slave MCC cards, to the master MCC by the MASTER FFT INIT signal from an upstream MCC. The PAL counters, 16G and 15G, count through the 516 states of the sequencer to address 516 memory locations of the four 7C245A 2K X 8 ROMs. The ROMs are programmed to provide timing control for many of the functions of the MCC.

Some of the control signals out of the sequencer ROMs are dedicated lines such as S12 (STA READ) out of pin 14 of ROM 18G. Other functions are encoded across several bits and decoded into individual timing signals by stages such as PALs 19F, 16E, or 20F. Some control terms are timing signals required on the MCC itself, like the ENA and RST signals coming out of PAL 14G, while other signals go off the card and provide the system with timing signals.

2.2.2 The Fringe/Integration Sequencer

The second sequencer on sheet 2 of the MCC schematic is the fringe-integrator cycle sequencer. PALs 16F and 15F count through the 8192 FFT cycles of an integration cycle while ROMs 17F and 18F provide timing signals during the course of this 131 msec cycle. The master MCC sequencer free runs and the slave cards use the INTEG CYC ENA signal for integration cycle synchronization (see the block diagram on the left hand border of sheet 2).

2.2.3 The MCC Microprocessor

The MCC microprocessor can be seen on sheet 2 of the MCC schematic. The 87C51 has a number of functions on the MCC card:

1) communicate via an HCB port with the RTS.

2) program the delay lines (14E, 17E, 18E, and 19E) that are set to insure good timing of 32-MHz signals to other bins.

3) provide the master/slave flag as instructed by the RTS (26J-12).

4) load the MAC card ASIC control words.

5) control the test frame generator.

The HCB support is seen in the lower right hand corner of the MCC schematic. The HCB communication is interrupt driven via the 87C51 INT1 interrupt.

2.2.4 Data Valid Muxes

Each FFT card in the VLBA correlator has a 4-into-1 data mux at the input of the card to select one of four possible PBD/DEF (Playback Drive / Deformatter) data sources. This mux function is duplicated on the MCC for the corresponding PBD/DEF validity signals.

A DEF card provides validity flags with each output channel that reflects the status of the data coming out of the DEF for that channel. These validity signals are updated on an FFT cycle basis. When improper operation (for example, a PBD frame with many parity errors) is detected in the transport output signal, the DEF uses the validity output to flag the FFT engine it feeds to discard all Fourier transform results polluted by data bits from the bad frame. In addition to discarding such invalid spectra, a count must be kept of how many such bad FFT cycles were discarded (actually, valid FFT cycles are counted).

The MCC card provides 4-into-1 muxes for the DEF validity outputs so a count can be made of the valid FFT cycles performed in a given FFT engine. Program words that address the 74ALS153s, seen on sheet 3 of the MCC schematic, are assembled by the RTS to duplicate the mux program bits on the FFT cards and stored in the registers 22E, 23E, 22C, 23C, and 22B.

DEF validity signals from two channels (all 24 DEF cards) go to a given MCC card. The capture signal CAPVA clocks the input registers for the validity signals once per FFT cycle and the output of the muxes go to registers (28E, 29E, 28C, 28B, and 28A), clocked once per FFT cycle by signal CAPVB (the capture signals are decoded in PAL 23B from the FFT sequencer signals).

The PALs 30E, 29C, 29B, 29A, 30A, 30C, and 30B form two independently programmable dual 20-into-1 muxes (4 outputs, total). Program bits S16, S17, S18, S19, and S20, from the FFT cycle sequencer, select one of the 20 validity signals to drive the DATAVALID AX line (and simultaneously selects the corresponding B-channel validity signal to drive the DATAVALID BX output). Program bits S21, S22, S23, S24, and S25 select A and B y-axis validities (DATAVALID BX and DATAVALID BY). The goal of this operation is to form baseline validities by logically ANDing the validities of stations X and Y for all 210 combinations of X and Y for the 20 stations (since if either station of a baseline is invalid, the baseline cross product is invalid). The S16 thru S20 control signals and the S21 thru S25 control signals will cycle through the 210 combinations of X and Y during the course of an FFT cycle as determined by the sequencer ROMs 19G and 20G. The logical AND occurs in ICs 2C and 3C on sheet 4 of the schematic.

2.2.5 Validity Delay Lines

The output of the 74ALS153 4-into-1 muxes on sheet 3 of the MCC schematic also drive the ten 27LS07 RAMs at the right edge of the page. The validity signals out of the muxes of the MCC must be connected to their respective FFT engines. It is in the FFT engine that the validity signals are acted upon (when the data being transformed is invalid ,

i.e. when the validity signal is at logic zero, the transform output is forced equal to zero).

The validity signal is connected to the ASIC of butterfly stage 4 in an FFT pipeline. A delay is provided on the MCC to duplicate the pipeline delay that data incurs in the FFT engine getting to stage 4 so that the data and validity signals that meet are from the same epoch. This delay is obtained in the 27LS07 RAMs (31E, 32E, 31D, and others). PAL 23B generates RAM addresses in a loop whose size reflects the delay required for the validity lines. A RAM address is selected, the validity signals previously stored in the RAM are read and new validity signals are written into the RAMs. The length of time until these RAM addresses are read again during the next address loop is the delay given to the validity signals. Each address of the address loop delays the validity signals an integral FFT cycle. Fractional FFT cycles are obtained by the phasing of the read and write signals within an FFT cycle.

The delayed validity output, from ICs 33E, 34E, 33C, 33B, and 33A, are wired into the FFT bins.

2.2.6 Multiplier Control Logic

The logic seen in the lower right hand corner of the MCC schematic, sheet 3, is the multiplier control logic. This logic generates control signals for the MAC cards in the bin the MCC occupies.

ICs 13B and 12B form a baseline counter that counts through 210 baselines (plus 3 special validity "baselines") in the course of reading out MAC results. The BLA[0..7] bus is the radix 210 (or, 213 as per above to be exact) counter output and ICs 12A, 11A, 8A, and 9A decode the BLA[0..7]baseline address into a MAC card select (CRDSEL\x) and row and column selects for the target ASIC chip on the MAC card. For each baseline, the MAC card has four results, 18-bit real and imaginary results from each of 2 channels. The two LSbits from IC 13B count through these four results.

The MAC control logic also provides the 8-bit EXT ADDR bus for the MAC cards. This bus is the source of the external address for the ASIC RAMs used for short term integration. This bus has three sources. IC 15B is the source of integration RAM addresses and counts through the 210 baselines in the course of an FFT cycle, IC 14B provides the external RAM address during short term integration readout, and IC 18B provides the external RAM address during self test readout. ROMs 17B and 16B are provided in case the RAM address sequence needs to be scrambled from a straight binary count. These ROMs were not used, however, and, as programmed, are straight through look-up tables. The only real function the ROMs serve is the 3-state output function.

IC 20B is the source of the bank bit of the MAC external RAM address. This signal toggles every 131 msec integration cycle to swap the integrating bank of the MAC ASIC RAM and the secondary storage (readout) bank of the RAM.

The control signals that drive more than one MAC card have more than one version for fanout purposes. Some signals have 2 fanout versions (such as AUXOUT SEL), some have 3 (such as the external address) and some have 4 (such as MACWE).

2.2.7 Data Validity and Pulsar Validity Counters

Sheet 4 of the MCC schematic is mostly the logic required for the tape data valid and the pulsar data valid counters. Each MCC handles the data validities for 2 channels (2 counters required) and the pulsar validities for one FCC card (4 counters required).

The function required of the validity counters is to duplicate the action of the MAC short term integrators, but whereas the MAC short term accumulators integrate astronomical cross products, the validity counters count the number of FFT cycles in an integration cycle for which a given baseline has valid data. For each baseline, the two stations of the baseline have their station data valids, as developed at the output of IC 30C or 30B (sheet 3 of the MCC schematic), logically ANDed together to produce a baseline valid signal (VA or VB from IC 2C or 3C). The pulsar validity counters count the number of FFT cycles, in an integration cycle, for which the pulsar gate (see the FCC card description) is permissive. Each spectral point has an individual counter.

A data valid counter consists of 210 binary counters (across 210 RAM addresses) each of which counts from 0 to 8192 and reflect, at the end of an integration cycle, the number of FFT cycles of the integration cycle for which a given baseline had valid data. A pulsar validity counter consists of 256 binary counters (one for each spectral point) each of which counts from 0 to 8192 and reflect, at the end of an integration cycle, the number of FFT cycles of the integration cycle for which a given spectral point was integrated. (Recall that the pulsar gate generator develops a signal used to retain or discard station spectral data, on a spectral point basis, as a function of the phase in the pulsar period. When the pulsar pulse is on, the pulsar gate will be expressed as permissive, allowing cross products to be integrated, when the pulse is off the gate will inhibit integration by forcing the station spectral point value to zero. Frequency dispersion causes the pulse to appear to be on in different spectral points at different times, hence the pulsar gate for different spectral points will be different.)

The 6 validity counters seen on sheet 4 of the MCC schematic, 2 for data and 4 for pulsar, are identical. RAMs 1D and 2D, latches 1E and 2E, and PAL adders 1F, 2F, and 2G constitute one validity counter. The RAMs can be thought of as being in parallel with the MAC ASIC short term accumulator RAMs. The RAM address counters 2C, 3C, and 4C are analog to the 3 sources of the EXT ADDR bus seen on sheet 3 of the MCC schematic (15B, 14B, and 18B). In short:

1) the RAM address counter addresses a running sum in the RAMs.
2) the partial sum from that RAM location is latched into the latch.

3) the PAL adder (for example, ICs 1F, 2F, and 2G) adds the logical state of a validity signal (for example DVA on latch pin 2E-18, applied to the adder on pins 1F-8, 2F-8, and 2G-8) to the old count (the old count is incremented if the validity signal is a logic one).
4) the new sum is written back into the same RAM address.

The control signals (from PALs 5B, 6B, 4B, 6C, and 11C) and the RAM address counters control the validity counters so that they operate step by step in parallel with the MAC short term accumulators. Output registers, for example ICs 1G and 1H, are analogous to the output registers in the MAC ASICs that store 18-bit output results. During the 4-bit gap between FFT cycles, results can be read out of the RAMs (using the output address counter 3C or 9C) and stored in these registers. The registers are subsequently read on the same 3-state bus as the MAC results (they are stored in the LTA as baselines 210, 211, and 212). PALs 6B and 11C provide two bits (always zero) during validity readout to bring the bus drive to 18-bits.

Both the data valids and pulsar valids must be delayed before being applied to the validity counters to compensate for the pipeline delays encountered in the data path (the MAC short term integrators and the validity counters should work in exact synchronism). The delay lines seen at the bottom of sheet 4 of the MCC schematic provide this delay. Two delay lines are seen, one for the 2 baseline data valid signals (ICs 3B, 2B, and 1C) and one for the 4 pulsar validity signals (ICs 7B, 8B, and 7C). Control logic for the delay lines is provided by IC 9B. Each delay line is a RAM clocked around a closed loop of addresses. The size of the loop defines the total time between a bit going in the RAM and the time it comes out and hence the delay obtained in the delay line.

2.2.8 MAC Output Bus

The MAC output bus, MAC BUS[0..17], passes through the MCC. This bus cycles through the 210 baseline results (baselines 0 through 209) from the short term integrators every 131 msec integration cycle. The validity counters described above are 3-stated onto this bus and enter the long term accumulator (LTA) as 3 additional "baselines" (210, 211, and 212). The bus drivers 2H, 4H, 6H, 8H, and 10H provide differential drive into the LTA.

2.2.9 Snapshot FIFO

As the short term results pass through the MCC on the way to the LTA, the FIFO stages (ICs 14I and 15I) can capture one set of 131 msec results for any given baseline. The desired baseline number is written into IC 18I by the microprocessor and the control logic of PAL 16I clocks 18-bit results into the FIFO every time the target baseline drives the MAC bus. The local micro then reads the FIFO contents. The 2 FIFO chips are tri-stated together with 8 of the 9-bits from each FIFO going directly to the microprocessor data bus and the 9th bit of the FIFOs being multiplexed together in IC 8C.

2.3 MCC software

The 87C51 microprocessor on the MCC card has a 32K X 8 main memory RAM. The main program for the MCC microprocessor is downloaded by the RTS into this memory.

2.3.1 Descriptions of the MCC Software

Below is a brief description of each of the modules listed in the chart below. (Memory allocations seen below are as of 9/17/92.)

Module Name	Starting Address	Ending Address	Memory
MASTER.ASM	0000	0149	ROM
ROMHCB.ASM	014A	02FE	ROM
ASIC.ASM	02FF	03AD	ROM
TEST.ASM	03AE	053C	ROM
TIME.ASM	053D	0659	ROM
MONITOR.ASM	0800	0F0F	ROM
RAM.ASM	1000	1023	RAM
RAMTEST.ASM	1024	1045	RAM
HELP.ASM	1046	1D9D	RAM
AUTOPLOT.ASM	1D9E	1E3B	RAM
RAMHCB.ASM	2000	22EB	RAM
OBS.ASM	22EC	2431	RAM
DISPLAY.ASM	3000	37EF	RAM
PLOT.ASM	4000	47E3	RAM
AUTOTEST.ASM	4800	4DEB	RAM

2.3.1.1 MASTER.ASM

MASTER.ASM has the software executed after a hardware reset to the microprocessor. This software will initialize the card and microprocessor functions like the interrupts, the serial port, etc. This module also has the interrupt vector locations. After a reset, the micro will execute from an idle loop in MASTER.ASM awaiting instructions from the terminal or RTS conveyed by interrupts. The fringe cycle interrupt is enabled in the idle loop so that the time code in the PBD frame will appear to run normally.

2.3.1.2 ROMHCB.ASM and RAMHCB.ASM

ROMHCB.ASM and RAMHCB.ASM handle the microprocessor communications with the RTS. 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.

2.3.1.3 ASIC.ASM

ASIC.ASM has the software to load the MAC ASIC control words. The table ATABLE contains the default ASIC control words loaded when a reset occurs.

2.3.1.4 TEST.ASM and RAMTEST.ASM

The TEST.ASM module has a number of hardware test routines that allow an operator to test the MCC 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 MCC ROMs.

2.3.1.5 MONITOR.ASM

The monitor software package has various terminal options supported by the MCC microprocessor.

2.3.1.6 RAM.ASM

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

2.3.1.7 HELP.ASM

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

2.3.1.8 AUTOPLOT.ASM

This module contains software to support the PLOT routine for plotting spectra captured by the FIFO. PLOT runs on a PC and can be used to view MAC short term accumulator results. The program on the PC is plot.exe and uses a serial port to connect to the MCC. The plot program is menu driven.

2.3.1.9 OBS.ASM

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

Once the card is configured, the MCC 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.

Actually, except for supporting self test, the MCC OBS.ASM program has very little to do. Support of the test frame time code and the FIFO readout occurs whether or not the MCC is observing.

2.3.1.10 DISPLAY.ASM

This module contains software to support the display of actual MAC results from the FIFO. It works through the monitor with the monitor command Vyxx. When evoked, this command will cause the FIFO to take a snapshot of the MAC result for baseline xx, channel y, and print the 15, 15, 6 results on the terminal. Each terminal screen will display a 4-by-16 array of numbers and the terminal keys 1, 2, 3, or 4 will choose from 4 possible screens. The screen display can be either single shot or will continuously update as new FIFO snapshots are taken depending on the monitor S function.

2.3.1.11 PLOT.ASM

This module contains software to support the plotting of actual MAC results from the FIFO (on the local terminal as opposed to PLOT, above, which interfaces through a PC). It works through the monitor with the monitor command Byxx. When evoked, this command will cause the FIFO to take a snapshot of the MAC result for baseline xx, channel y, and do a very crude plot of the spectrum. The screen will continuously update as new FIFO snapshots are taken (unless the monitor S function prohibits).

2.3.1.12 AUTOTEST.ASM

AUTOTEST.ASM is another test routine that uses the FIFO on the MCC card. It is run through the monitor and has the monitor command Qyxx. When evoked, it will first take a snapshot of baseline xx, channel y, and store the results for use as predicted data for the rest of the test. The program will then cycle through all 210 baselines, one at a time, taking snapshots and checking the resulting spectrum against the predicted snapshot. A terminal screen is produced that gives the count of times a given baseline snapshot differed from the predicted. The count saturates at hex F.

If the Qyxx command is used with xx = D2 (the validity baseline number), a different function results. Now each of the 210 results comes down to a single value instead of 256 spectral points and each point is checked against a few specific values. The test result should be self-explanatory. (When xx = d3 or d4, the pulsar validity baseline numbers, the result is different again.)

2.3.2 Microprocessor Interrupt Structure

Almost all of the functions provided by the MCC software are done in interrupt routines. There are several interrupts and a summary is given below:

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. This interrupt is generated by the sequencers and actually occur 4 times per 4.128 msec fringe cycle. The MCC microprocessor senses the T1 micro input to get integration cycle synchronization and also to initialize a software counter so it can distinguish which of the 4 interrupts per fringe cycle is the actual fringe cycle indicator.

The reason for having 4 interrupts per fringe cycle is so the microprocessor can update the PBD test frame time. The PBD frame period is 2.5 msec and the microprocessor has no interrupt to allow it to know when to update the frame time. The micro instead polls the HEADER discrete in the fringe cycle interrupt routine to know when a header has occurred hence when the time can be updated. The 4.128/4 msec interrupt is frequent enough so it will never miss the time update requirement.

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

2.4 MCC Utility Displays

The MCC terminal provides various utility functions for use when investigating problems in the system. At the MCC prompt a HELP screen is available:

MCC HELP screen (invoke at MCC prompt with letter H):

```
DXXXX       DISPLAY STARTING AT MEM LOC XXXX
PXXXX       DISPLAY PAGE STARTING AT MEM LOC XXXX
C/R         DISPLAY MORE (D, T, P, OR H)
D           DISPLAY STARTING AT LAST MEM LOC XXXX
P           DISPLAY PAGE STARTING AT LAST MEM LOC XXXX
R           DISPLAY INTERNAL RAM
RXX YY      MODIFY INTERNAL RAM
MXXXX       MODIFY STARTING AT MEM LOC XXXX
GXXXX       GOTO MEM LOC XXXX
LZZ         STORES FIFO FULL OF BASELINE ZZ DATA STARTING AT LOC 7000
VYZZ        DISPLAY RESULTS, BASELINE ZZ, CHANNEL Y
BYZZ        PLOT SPECTRA FOR BASELINE ZZ, CHANNEL Y
QYZZ        AUTOTEST, USE BASELINE ZZ, CHANNEL Y, FOR PREDICTED
                 ZZ = 00 TO D1, GETS BASELINE 00 TO 209
                 ZZ = D2, GETS DATA VALIDS (PREDICTED IN RAM LOC 50-51)
                 ZZ = D3, GETS PULSAR DATA VALIDS
                 ZZ = D4, GETS PULSAR DATA VALIDS
                 ZZ = FF, NO NEW PREDICTEDS TAKEN


                 ,C/R FOR NEXT HELP SCREEN
```

Second part (invoke with C/R):

```
FX          SET PLL FREQ FOR STANDARD FREQ (X=0 9.001674 MHZ)
FXXX        SET PLL FREQ XXX
HX          SET HEADER RAM CONTENTS
HXX YY      MODIFY HEADER RAM STARTING AT LOC XX
IXX YY      SET HEADER TIME INCREMENT
TX          EXECUTE TEST X
TX YY       ANOTHER FORM OF TEST
TM          TEST RAM MEMORY
WXX         FILL TAPE FRAME RAM WITH XX BYTE
H OR ?      (OR HELP) HELP SCREEN
A*          PERFORM LAST INSTRUCTION AGAIN
C*          DISPLAY VALIDITY CROSS-BAR SWITCH
S*          SWITCH BETWEEN INTERNAL AND EXT MEM (DISPLAY & PAGE)
X*          TOGGLE RANDOM DATA/RAM DATA IN TAPE FRAME
Y*          TOGGLE VLBA/MKIII TAPE FRAME FORMAT
Z*          TOGGLE TIME IN HEADER RUNNING/NOT RUNNING CONDITION
.*          CLEAR CRT SCREEN

                 ,C/R FOR NEXT HELP SCREEN
```

Third part:

```
T0          SENDS A BINARY COUNT TO CHIP 28J
T1          SENDS A BINARY COUNT TO CHIP 28I
T2          SENDS A BINARY COUNT TO CHIP 25I
T3          SENDS A BINARY COUNT TO CHIP 26J AND 26I
T4 X        SENDS A BINARY COUNT TO CHIP 22E, 23E, 22C, 23C, 22B, OR 21C
T5          SENDS A BINARY COUNT TO CHIP 18I
T6 XX       TESTS THE CHIP PROG WORD LOGIC (CARD GIVEN BY XX)
T7          WRITES AND CHECKS THE FRAME BUFFER RAM (29F)
T8          WRITES AND CHECKS THE HEADER BUFFER RAM (27F)
T9          WILL TEST THE PLL
TA          LOOP ON TEST 9
TB
TC
TD
TE
TF          TEST IN RAM

TF 00        LOOP SENDING ASIC CONTROL WORDS

                ,C/R FOR NEXT HELP SCREEN
```

Fourth part:

```
BYXX        PLOT SPECTRA FROM BASELINE XX, CHANNEL Y
                1* TO 7*            SELECT HORIZONTAL SCALE
                E*, C*              EXPAND, COMPACT VERTICAL SCALE
                U*, D*              MOVE PLOT UP, DOWN
                R*, I*, OR M*       WILL GIVE REAL, IMAGINARY, OR MAGNITUDE PLOT

VYXX        DISPLAY RESULTS, BASELINE XX, CHANNEL Y
                1* TO 4*            SELECT PAGE 1 TO PAGE 4

QYXX        AUTOTEST OF CHANNEL Y (TAKES PREDICTED SNAPSHOT FROM BASELINE XX)
                Z*                  ZERO ERRORS

B,V,OR Q    COMMON OPTIONS
                N*                  NEXT BASELINE
                X*                  SWITCH CHANNEL
                S*                  SWITCHES FROM CONTINUOUS TO SINGLE SHOT
                CR                  TAKES ANOTHER SHOT IN SINGLE SHOT MODE
                SP OR ESC           EXITS PLOT ROUTINE

LXX         STORES FIFO FULL OF BASELINE XX DATA STARTING AT LOC 7000
                S*                  SWITCHES FROM CONTINUOUS TO SINGLE SHOT
```

Examples of two useful commands are shown below:

The C command for the cross bar assignment displays as follows:

ANTENNA	MUX SELECTION
00	0
01	0
02	0
03	0
04	0
05	0
06	0
07	0
08	0
09	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	3
18	0
19	0

TEST = 00 COUNT = 05

At each four into one mux, the MUX SELECTION column shows which of the four inputs is selected; these tape validity muxes on the MCC must be programmed the same as the tape data muxes on the FFT cards.

The V0D2 command shows validity values for baseline 0xD2 (210 decimal): (SPACE bar ends the updating of the display)
This example could apply to a job with stations 0-12 in use. The E indicates a validity count equal to the 0x2000 stored for reference at ram locations 50-51. The B indicates a validity count in the range of 0x1F00 to 0x2000.

DATA VALIDITY D2 CHANNEL 4

ANT	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	E	E	E	E	E	E	E	B	E	E	E	E	E
1	0	0	0	0	0	0	0	E	E	E	E	E	E	E	B	E	E	E	E	
2	0	0	0	0	0	0	0	E	E	E	E	E	E	E	B	E	E	E		
3	0	0	0	0	0	0	0	E	E	E	E	E	E	E	B	E	E			
4	0	0	0	0	0	0	0	E	E	E	E	E	E	E	B	E				
5	0	0	0	0	0	0	0	B	B	B	B	B	B	B	B					
6	0	0	0	0	0	0	0	E	E	E	E	E	E	E						
7	0	0	0	0	0	0	0	E	E	E	E	E	E							
8	0	0	0	0	0	0	0	E	E	E	E	E								
9	0	0	0	0	0	0	0	E	E	E	E									
10	0	0	0	0	0	0	0	E	E	E										
11	0	0	0	0	0	0	0	E	E											
12	0	0	0	0	0	0	0	E												
13	0	0	0	0	0	0	0													
14	0	0	0	0	0	0														
15	0	0	0	0	0															
16	0	0	0	0																
17	0	0	0																	
18	0	0																		
19	0																			

E = RAM 50-51 VALUE
* = >2000 (ILLEGAL)
1 = 2000
B = (2000,1F00]
M = (1F00,1000]
m = (1000,0100]
S = (0100,0001]
0 = 0000

The following table tabulates the number of invalid FFT cycles per 4 msec tick due to buffer activity in the DEF (assuming low bandwidth mode is not in use). This table is excerpted from stnTask.c. It can be used to calculate the expected number of valid FFT cycles in 131 msec (the V0D2 display above is based on the number of valid FFT cycles in 131 msec). For example, there are 256 FFT cycles per 4 msec, and 32 4 msec cycles per 131 msec, so the maximum possible validity count in 131 msec is 32 * 256 = 8192 = 0x2000.

```
 1    2    4    8   16  ← Oversample factors
--   --   --   --   --

/* FFT size 64 */
 0,   0,   2,   6,  14,        /* 1-4 */
 1,   2,   6,  14,  30,        /* 1-2 */
 3,   6,  14,  30,  64,        /* 1-1 */
 7,  14,  30,  64, 256,        /* 2-1 */
15,  30,  64, 256, 256,        /* 4-1 */

/* FFT size 128 */
 0,   0,   2,   6,  14,        /* 1-4 */
 1,   2,   6,  14,  30,        /* 1-2 */
 3,   6,  14,  30,  64,        /* 1-1 */
 7,  14,  30,  64, 128,        /* 2-1 */
15,  30,  64, 128, 256,        /* 4-1 */

/* FFT size 256 */
 0,   0,   2,   6,  14,        /* 1-4 */
 1,   2,   6,  14,  30,        /* 1-2 */
 3,   6,  14,  30,  64,        /* 1-1 */
 7,  14,  30,  64, 128,        /* 2-1 */
15,  30,  64, 128, 256,        /* 4-1 */

/* FFT size 512 */
 0,   0,   2,   6,  14,        /* 1-4 */
 1,   2,   6,  14,  30,        /* 1-2 */
 3,   6,  14,  30,  64,        /* 1-1 */
 7,  14,  30,  64, 128,        /* 2-1 */
14,  30,  64, 128, 256,        /* 4-1 */

/* FFT size 1024 */
 1,   2,   6,  14,  30,        /* 1-4 */
 3,   6,  14,  30,  64,        /* 1-2 */
 7,  14,  30,  64, 128,        /* 1-1 */
15,  30,  64, 128, 256,        /* 2-1 */
32,  64, 128, 256, 256,        /* 4-1 */

/* FFT size 2048 */
 3,   6,  14,  30,  64,        /* 1-4 */
 7,  14,  30,  64, 128,        /* 1-2 */
15,  30,  64, 128, 256,        /* 1-1 */
32,  64, 128, 256, 256,        /* 2-1 */
64, 128, 256, 256, 256         /* 4-1 */
```

2.5 The MCC PLOT Program

The MCC terminal can also be used to display amplitude and phase plots of the spectra for any baseline. The executable program used for this is named plot.exe and is usually run by first connecting to the MCC using the VTERM terminal emulator to verify that a valid MCC prompt is operating, then escape from VTERM using the double shift key operation. At the DOS prompt, invoke the PLOT program. The program prompts for the baseline (in hex), the channel (0 or 1) and the desired display (amplitude or phase etc.).

The plot.exe and plot.c files are presently stored in the CORRDWGS area in the directory /misc/re/c. Copies of plot.exe are available on the four PCs in the correlator area and on the PC used with the testbed rack.

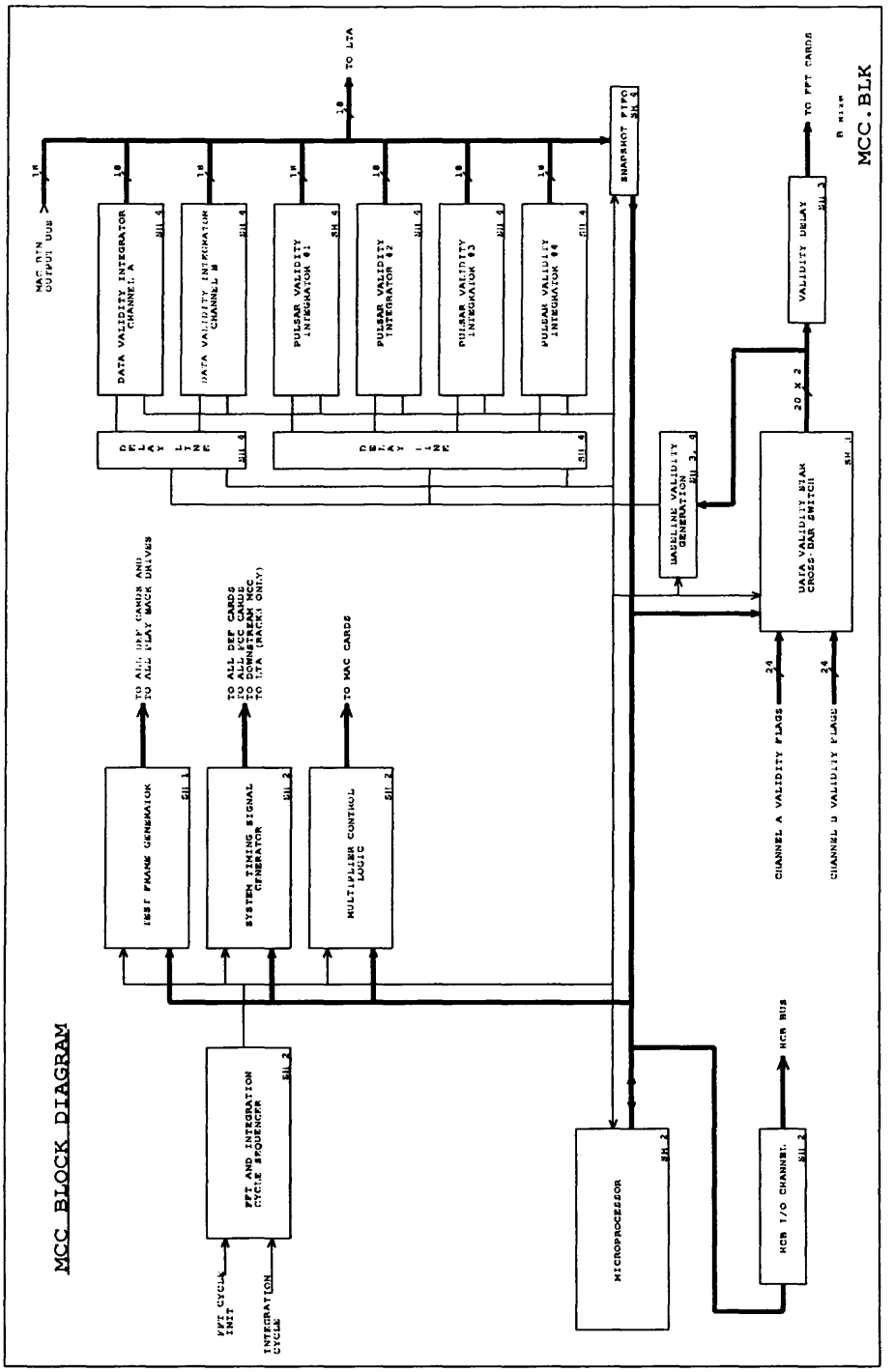


Figure 3 Master Control Card Block Diagram

3 The Multiply and Accumulate Card (MAC)

3.1 Introduction

The MAC Cards (also referred to as Multiplier Cards) perform the multiplication and accumulation functions in the VLBA Correlator. Both self and cross multiplication products are done for 20 stations times 20 stations, producing 210 baselines for each of 8 baseband converter channels, in arrays of MAC cards.

The MAC has two modes of operation. These modes are non-polarization (MACNP) and polarization (MACP). In non-polarization mode, the multipliers can be logically viewed as 8 distinct triangular arrays of 210 baselines each.

In order to handle polarization mode, pairs of arrays are physically combined, forming four combined arrays as seen in the FFT to MULTIPLIER INTERFACE block diagram, K029D01.BLK, found in Volume 2 of this report. Polarization pairs are handled in adjacent FFT pipelines, and the output data streams are multiplexed together. The multiplexed data streams drive VLBA1 ASICs in both the even and odd sections of the combined arrays.

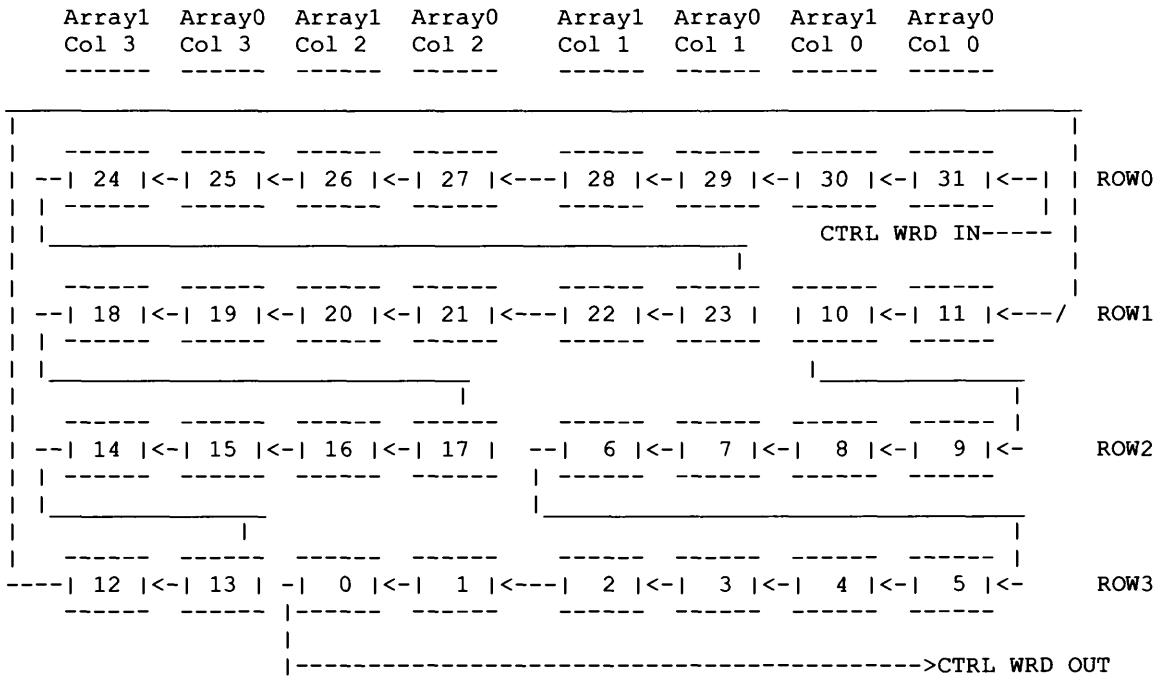
In non-polar mode, the even sections of the arrays produce the Right x Right (RxR) products and the odd sections produce the Left x Left (LxL) products, by processing the appropriate FFT results in the multiplexed data streams. In polar mode, the even sections produce RxR and RxL while the odd sections produce LxR and LxL, as shown in K029D01.BLK.

A single combined array consists of 15 MAC cards, numbered 0 through 14 as seen in K029D01.BLK. See Figure 2 for the layout of the 14 cards in a MAC bin. The cards located on the array diagonal are cards number 0, 5, 9, 12 and 14. The cards on the diagonal are partially populated in order to not have unused VLBA1 chips running under power in un-used locations (below the diagonal). In Figure 2, these card locations are indicated with an asterisk.

The block diagram of a single Multiplier Card is shown in K030D01.BLK, found in Volume 2 of this report. The physical layout of the VLBA1 chips on the card is in two square arrays of 16 chips each, but K030D01.BLK shows a logical arrangement rather than the physical layout. The physical layout can be seen on sheet one of the MAC schematic, L003D01.SCH (see Volume 2). In both K030D01.BLK and L003D01.SCH, the VLBA1 chips below the dashed lines are the chips that are not installed in the partially populated cards located on the array diagonal.

3.1.1 The MAC VLBA1 Control Word Daisy Chain

The 32 VLBA1 chips on a MAC card are daisy chained together in such a way that partially populated cards along the diagonal of the multiplier array may be loaded with VLBA1 control words using the same software as a fully populated card. This requires that the control word daisy chain connects to all of the VLBA1 chips "on and above the diagonal" before connecting to the chips below the diagonal.



The offset (OFS) within the daisy chain is indicated for each VLBA1 location, where offset 0 is the first 32 bit control word shifted into the card and offset 31 is the last control word shifted in.

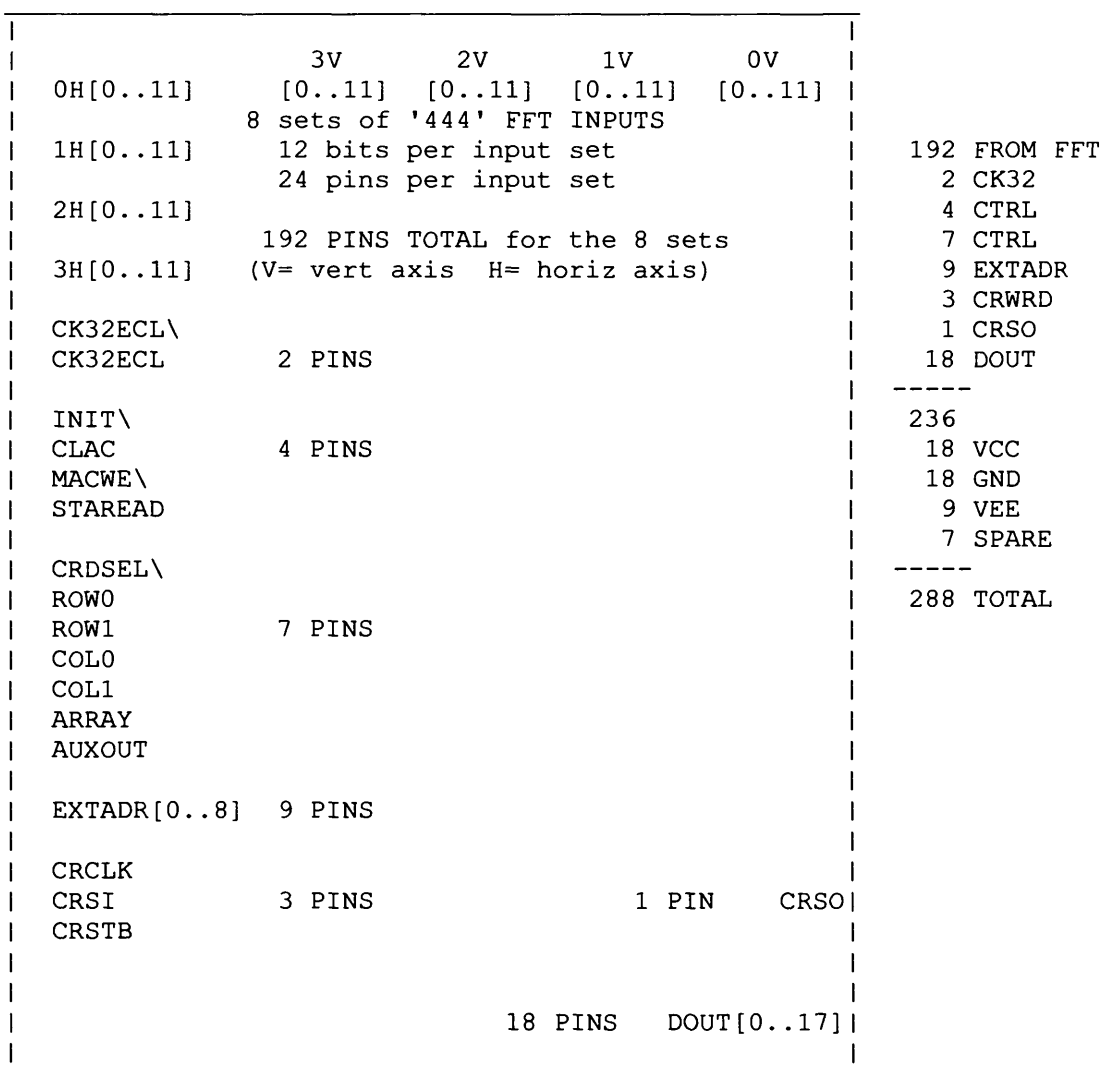
OFS	ROW	COL	ARRAY		OFS	ROW	COL	ARRAY
0	3	2	1	<-----FIRST WORD	16	2	2	1
1	3	2	0	SHIFTED INTO	17	2	2	0
2	3	1	1	CARD	18	1	3	1
3	3	1	0		19	1	3	0
4	3	0	1		20	1	2	1
5	3	0	0		21	1	2	0
6	2	1	1		22	1	1	1
7	2	1	0		23	1	1	0
8	2	0	1		24	0	3	1
9	2	0	0		25	0	3	0
10	1	0	1		26	0	2	1
11	1	0	0		27	0	2	0
					28	0	1	1
12	3	3	1		29	0	1	0
13	3	3	0		30	0	0	1
14	2	3	1		31	0	0	0
15	2	3	0	(last word shifted into card)				

3.2 MAC Card Description

The MAC card logic is composed primarily of the VLBA1 ASIC chips, with a minimum of glue logic required. The data streams from the FFT cards are in 4,4,4 format (four bits of real mantissa, four bits of imaginary mantissa, and four bits of common exponent). These input data signals are all differential ECL. These inputs are converted to TTL using 10125 ECL to TTL translators, and then captured by the 32 MHz system clock in 74F821 or 74F574 registers.

Refer to the schematic sheets L003D01.SCH through L003D04.SCH, found in Volume 2, for the following descriptions.

3.2.1 MAC Card I/O Pin Definitions



Signal Descriptions:

FFT INPUTS:
'444' signals from FFT engines; 32 MHz rate, ECL differential; 6 bridging loads per run, 100 ohm termination at end of run on backplane;
each '444' signal consists of bits 0-11 (12 bits total);
8 sets total * 12 bits per set * 2 pins per bit = 192 pins;

CK32 inputs: SYSTEM CLOCK
differential ECL master clock to card, 2 pins; 100 ohm termination on card;

INIT\: FFT CYCLE INIT PULSE
TTL single ended, 32 MHz resolution, drives one 74F574 input on each card;
one clock pulse wide, asserted once per fft cycle

CLAC: CLEAR ACCUMULATOR
TTL single ended, 32 MHz resolution, drives one 74F574 input on each card;
asserted during first cycle of accumulation into a bank of ram;

MACWE\: MAC WRITE ENABLE
TTL single ended, 32 MHz resolution, drives one 74F574 input on each card;
Asserted during bit period #514 (periods= 0-515); causes off line ram bank contents to be strobed into 36 bit storage register on the ASIC (the external address bus applies the ram address during bit period #513); if CLAC happens to be asserted during the MACWE\ pulse, the data will be blanked;

STAREAD: SHORT TERM ACCUMULATOR READ
TTL single ended, 32 MHz resolution, drives one 74F574 input on each card;
defines the read out cycle for reading the 36 bit results in the storage register on the ASIC; STAREAD is high for one 32 MHz clock period, low for 7 clock periods;
the rising edge of STAREAD captures the "address" (CRDSEL\, ROW, COL, ARRAY and AUXOUT) of the result to be read; if a result is selected during one STAREAD cycle, that result is valid at the card output during the low portion of the next STAREAD cycle;

CRDSEL\ CARD SELECT
ROW[0..1] ROW SELECT
COL[0..1] COLUMN SELECT
ARRAY ARRAY SELECT
AUXOUT: 18 BIT CHUNK SELECT
TTL single ended, 4 MHz resolution, drive one 74F574 input on each card;

When a hardware baseline result from an array is to be selected, the MCC translates the baseline number, 0-209, into a card select, row select and column select in order to access a single ASIC in an array on a MAC card.

When AUXOUT is logic zero, the 18 bits selected from the ASIC RAM B are:

d17	d16	d15	d14	d13	d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0
ex5	ex4	ex3	i14	i13	i12	i11	i10	i9	i8	i7	i6	i5	i4	i3	i2	i1	i0

When AUXOUT is logic one, the 18 bits selected from the ASIC RAM A are:

d17	d16	d15	d14	d13	d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0
ex2	ex1	ex0	r14	r13	r12	r11	r10	r9	r8	r7	r6	r5	r4	r3	r2	r1	r0

where the data is 15,15,6 format:

15 bits of real mantissa, one's complement format
15 bits of imag mantissa, one's complement format
6 bits of common exponent, two's complement format

EXTADR[0..8]: EXTERNAL ADDRESS
TTL single ended, 16 MHz resolution, drive one 74F824 input on each card;

Provides the READ address for ASIC accesses; the internal address generator provides the WRITE address;

CRCLK CONTROL REGISTER CLOCK
CRSI CONTROL REGISTER SERIAL DATA INPUT
CRSTB CONTROL REGISTER STROBE
CRSO: CONTROL REGISTER SERIAL DATA OUTPUT
TTL single ended, lower resolution (87C51 driver) inputs drive one 74F541 input on each card;

CRCLK rising edge clocks 32 serial data bits (bit 0 first) into each ASIC; ASICs are daisy chained on a card; CRSTB pulses high to latch the serial register contents into a secondary storage latch;

DOUT[0..17]: DATA OUTPUT
TTL tri-state card output bus; outputs of 15 cards tri-stated together within a chassis;

In both MACNP and MACP modes, the INIT, EXTADR, CLAC and MACWE signals require a single 32 MHz clock cycle delay for the versions that go to array 1, with respect to the array 0 inputs.

3.2.2 FFT to MAC Interface

Adjacent channel 4,4,4 results from the FFT are multiplexed together. For purposes of discussion, even channels are assumed to be right polarization, odd channels left polarization, as seen in the FFT to MULTIPLIER INTERFACE block diagram, K029D01.BLK, found in Volume 2.

At the beginning of an FFT cycle, the first 4,4,4 data point coming out of CH0 is sent across the interface in one clock cycle, followed by the first point out of CH1. The data rate out of the FFT pipelines is 16 MHz, the data rate over the interface is 32 MHz.

At the FFT end, the multiplexing is done in a 2:1 mux with storage. The resulting signals are translated to ECL (10124) and sent differentially to the MAC cards. Each signal has 6 bridging loads, with a termination at the end of the run.

At the MAC end, each bridging load is a 10125 ECL/TTL translator, the output of which is captured in a 74F821 or similar register. Each register for vertical axis signals drives 8 ASIC inputs. Each register for horizontal axis signals drives 8 ASIC inputs. (In MAC modes, the NORM input is internally connected to the AUX input for use in MACPA1.)

3.3 Ram Addressing

The original scheme for addressing the rams in MAC mode called for a single system wide external address that would control both reading and writing during accumulation, and readout access for dumping the accumulated results.

The advent of sub-arrays, which allows MACP and MACNP to co-exist, made this impossible. The following list shows the address sequences required for both modes (see V009D01.TIM in Volume 2):

MACNP		MACP		
READ	WRITE	READ	WRITE	
0		0		These sequences are all
RD0		RD0		16 MHz clock rate counter
1		RD1		outputs.
RD1		1		
2		2		The "RD" and "WR" are used
RD2		RD2		to indicate when the address
3	WR0	RD3	0	must be present. The other
RD3	0	3	WR0	times were then filled in to
4	WR1	4	WR1	show that the same sequence
RD4	1	RD4	1	may be used for reading the
5	WR2	RD5	2	ram in both modes.
RD5	2	5	WR2	
6	WR3	6	WR3	
RD6	3	RD6	3	
7	WR4	RD7	4	

Since one read sequence can handle both modes, the system wide external address bus may be used for reading the rams during accumulation, as well as accessing the accumulated results when reading out the "off line" memory bank.

The external address can not also be used for the write address sequence, so the internal address generator is used for this purpose.

3.4 Card Edge Timing and MACWE Accesses

The timing relations between the 4,4,4 input data streams and the control signals (INIT, EXTADR and MACWE) at the MAC card edge are shown in drawing V009D01.TIM, found in Volume 2. This drawing starts with the required timings at the ASIC input pins in both array 0 and array 1, and then derives the card edge timing.

The four cycle gap (clock cycles #512 - #515) allows time to access results from the off line bank in the ASIC. Every 131 msec the ram bank in the ASIC switches. While one bank is accumulating, the other bank is available for readout. The ACCESS references in V009D01.TIM show the clock cycles at which the MACWE pulse and the EXTADR address must be present in order to read out a single result from the off line bank. These MACWE accesses occur every 27 FFT cycles, requiring 6912 FFT cycles to read out all 256 spectral results.

3.5 Short Term Accumulator Readout Cycle

After every MACWE access, the single 36 bit results are read from every MAC ASIC in the system, 18 bits at a time. The STAREAD (Short Term Accumulator Read) signal controls the readout. The rising edge of STAREAD clocks the "address" into U35. If CRDSEL\ is asserted (low), then the selected ASIC is output enabled onto one of the two 18 bit buses. See K030D01.BLK, Multiplier Card Block Diagram, in Volume 2, for the steps involved in selecting a MAC card, selecting an ASIC on the card and enabling the card output onto the backplane tri-state bus.

3.5.1 Readout Sequence

The PROM at location 12A on the MCC controls the readout sequence for reading the 36 bit results from each of the 210 baselines in the combined array controlled by a MCC. The PROM is addressed to sequence from baseline 0 through baseline 209. The outputs of the PROM are decoded to select a row, column and card. The following figure defines the baselines (0-209) on each card (0-14):

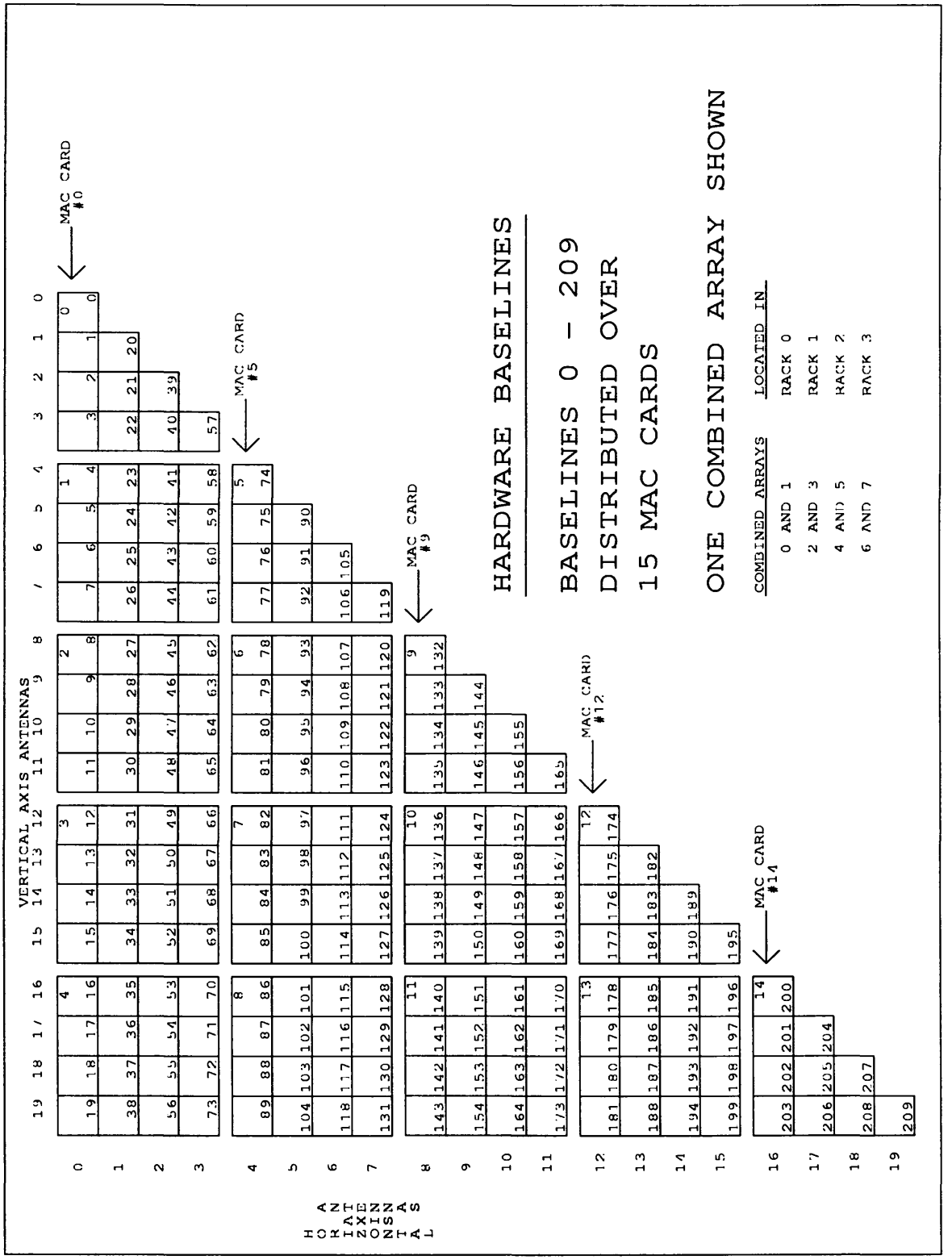


Figure 4 Hardware Baselines

The following three figures summarize the card numbers, column numbers and row numbers as a function of baseline. This information is required in the MCC PROM at location 12A.

Figure 5 Map of Cards versus Baselines

		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										VERTICAL											
0		4	4	4	4	3	3	3	3	2	2	2	2	1	1	1	1	0	0	0	0
1		4	4	4	4	3	3	3	3	2	2	2	2	1	1	1	1	0	0	0	|
2		4	4	4	4	3	3	3	3	2	2	2	2	1	1	1	1	0	0	|	|
3		4	4	4	4	3	3	3	3	2	2	2	2	1	1	1	1	0		|	|
4	H	8	8	8	8	7	7	7	7	6	6	6	6	5	5	5	5			|	|
5	O	8	8	8	8	7	7	7	7	6	6	6	6	5	5	5	baseline			|	|
6	R	8	8	8	8	7	7	7	7	6	6	6	6	5	5		nr 20				|
7	I	8	8	8	8	7	7	7	7	6	6	6	6	5							|
8	Z	11	11	11	11	10	10	10	10	9	9	9	9								|
9	O	11	11	11	11	10	10	10	10	9	9	9							baseline		|
10	N	11	11	11	11	10	10	10	10	9	9								nr 0		
11	T	11	11	11	11	10	10	10	10	9											
12	A	13	13	13	13	12	12	12	12												
13	L	13	13	13	13	12	12	12													
14		13	13	13	13	12	12														
15		13	13	13	13	12															
16		14	14	14	14																
17		14	14	14																	
18		14	14																		
19		14---------baseline nr 209																			

Figure 6 Map of Column Numbers versus Baselines

		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										VERTICAL											
0		3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
1		3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	
2		3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2		
3	H	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3			
4	O	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0				
5	R	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1					
6	I	3	2	1	0	3	2	1	0	3	2	1	0	3	2						
7	Z	3	2	1	0	3	2	1	0	3	2	1	0	3							
8	O	3	2	1	0	3	2	1	0	3	2	1	0								
9	N	3	2	1	0	3	2	1	0	3	2	1									
10	T	3	2	1	0	3	2	1	0	3	2										
11	A	3	2	1	0	3	2	1	0	3											
12	L	3	2	1	0	3	2	1	0												
13		3	2	1	0	3	2	1													
14		3	2	1	0	3	2														
15		3	2	1	0	3															
16		3	2	1	0																
17		3	2	1																	
18		3	2																		
19		3																			

Figure 7 Map of Row Numbers versus Baselines

		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										VERTICAL											
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2		
3		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3			
4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
5	H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					
6	O	2	2	2	2	2	2	2	2	2	2	2	2	2	2						
7	R	3	3	3	3	3	3	3	3	3	3	3	3	3							
8	I	0	0	0	0	0	0	0	0	0	0	0	0								
9	Z	1	1	1	1	1	1	1	1	1	1	1									
10	O	2	2	2	2	2	2	2	2	2	2										
11	N	3	3	3	3	3	3	3	3	3											
12	T	0	0	0	0	0	0	0	0												
13	A	1	1	1	1	1	1	1													
14	L	2	2	2	2	2	2														
15		3	3	3	3	3															
16		0	0	0	0																
17		1	1	1																	
18		2	2																		
19		3																			

3.6 *Tri-State Buses on the MAC Card*

As shown on K030D01.BLK, Multiplier Card Block Diagram, in Volume 2, there are two tri-state ASIC output buses on the MAC card. There are 16 VLBA1 ASICs on each bus, with columns 0 and 1 on one bus and columns 2 and 3 on the second bus. Both buses are switched to the high-Z state when STAREAD is high (at U36 pin 4 and U37 pin 4), to allow time for one ASIC to turn off before another turns on. There are no pullups on the bus, so the bus floats during this one 32 MHz clock period when the card is actually selected for readout. When the card is not selected, both buses are high-Z all the time. The 74LS157 two into one multiplexers at U54 - U58 have ambiguous inputs in these cases. When the card is not selected, the multiplexers are disabled, so the outputs are logic zero, independent of the inputs. When the card is selected, the outputs can respond to the ambiguous inputs for the single 32 MHz clock period. It has been observed that slight system test errors can occur, apparently as a result of the high-Z buses. These errors were observed at MAC SN 55 when installed at Rack 0, MAC Card # 12, usually just when the rack temperature was higher than normal. It was found that 74ALS157 type chips were used in all but the prototype MAC card and that replacing the ALS with LS prevented the errors from occurring on SN 55 (and on SN 49 when it was tried in the same slot). This change has not been done to all cards, but may be required on some additional cards in the future, if more of these errors are found. This problem was not seen to be serious enough to require the addition of pullups on the buses, and at this time no tests have been done to verify that pullups indeed would stop the errors.

(We attempted to reproduce the problem at a later date in order to test pullups, but did not succeed in causing the errors.

This page intentionally left blank.

4 THE LONG TERM ACCUMULATOR CARD (LTA)

4.1 Introduction

The LTA card accepts data from the MAC cards every 131 msec, to be accumulated for one or more 131 msec intervals before being transferred to the FIR card.

In a single 131 msec dump from the Multiply/Accumulate section (MAC) to the Long Term Accumulator (LTA), there are 430,080 results to be transferred (256 results per ASIC, 210 ASICs per array, 8 arrays = 430,080). Note that there are 8 MAC arrays (the 8 arrays of cross multipliers) and that each array is associated with a corresponding FFT pipeline channel. The MAC results are 15,15,6 format (15 bits of real mantissa, 15 bits of imaginary mantissa and 6 bits of exponent).

The 15,15,6 format MAC results are converted to IEEE single precision numbers before accumulation.

4.2 LTA General Description

The original system specifications called for the LTA to store 1024 results per baseline (1024 x 210 = 215,040 total results). Thus at a minimum we would have needed to reduce the available 430,080 results by a factor of two as they accumulate into the LTA. The final LTA design used eight 1 Meg deep by 8 bit wide DRAM modules, for a total ram size of 1 Meg by 64 bits wide. This provides storage for 2048 results per baseline. Data reduction still may be done using one or more of three methods:

> For spectral point averaging, adjacent spectral points may be accumulated into the same LTA locations.
>
> For FFT sizes of 256 (128 spectral points) or less, where there are multiple transforms in a single FFT pipeline at the same time, "adjacent" transforms may be accumulated together.
>
> Where different arrays have data that can be accumulated together, such as when doing overlapping in two or more FFT engines, we can accumulate results from 2, 4 or 8 arrays into the same LTA locations. This mode of operation has not been used so far.

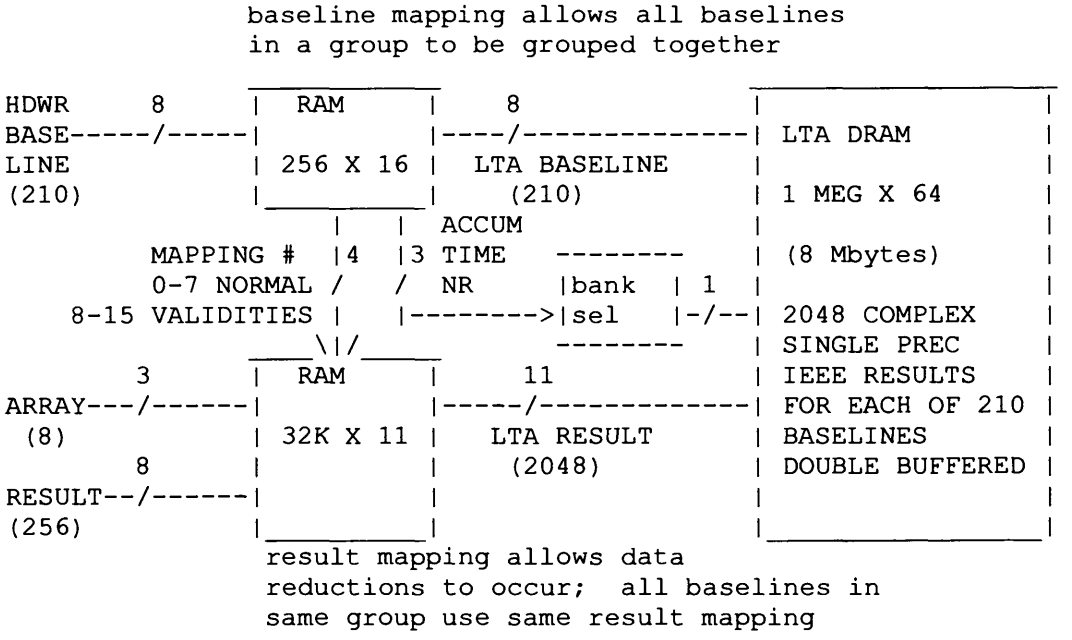
In order to accumulate 430,080 results in 131 msec (the integration cycle), we must access the ram 860,160 times. In 131 msec, we thus have an average of 152 ns per ram access, or 4.9 32 MHz system clock periods available. As discussed below, the sequence of ram accesses works out to 3.8 32 MHz system clock periods average per

accumulation time. The MS bit of the four bit mapping code is used to distinguish normal results from tape and pulsar validity results. Thus there are at most eight distinct input result mappings for normal results (and eight for validities), and there are at most eight distinct accumulation times

Each baseline group may bank switch on any multiple of integration cycles. At each bank switch, a Clear Accumulator (CLAC) function is required so that the new results coming in are added to zero rather than to the current contents of the storage location. When no forms of data reduction are occurring, then the CLAC asserts for the duration of the first 131 msec interval of the accumulation interval. This CLAC is the master CLAC controlled by the 87C51 and stored in an eight bit register (one bit for each accumulation code affiliation). When data reduction is occurring, the master CLAC must be gated with a CLAC Inhibit bit. This is due to the fact that more than one MAC result will be accumulated into a single LTA storage location. We thus require that CLAC be asserted only for the first MAC result and not be asserted for subsequent MAC results that will be summed into the same location. The Front End Result Translation Ram Table is the source of this CLAC Inhibit bit. The MS bit of the table is zero to enable the master CLAC to be applied. This bit is set to one to inhibit the master CLAC from being applied. Since this table has a separate entry for every individual MAC result (2048 results, 8 arrays of 256 results each), it should be possible to accommodate all possible combinations of the three data reduction types.

The following figure details the address mapping for visibilities input to the LTA ram. Except for the Fast Page Mode restriction mentioned above, the mapping from one of 2048 MAC results to one of 2048 LTA results is completely flexible for each of up to 8 mapping codes. The upper half of the result mapping ram is used for mapping validities, and is normally a straight thru table.

Figure 9 LTA INPUT MAPPING



The figure above does not show the details of bank selection for validities. A more detailed representation is found in the LTA Block Diagram, L024D06.DOC, found in Volume 2.

4.2.2 Backend Access

A backend access to a complex single precision IEEE result may occur approximately every 64 system clocks (64 x 31.25 ns = 2 us). Specifically, there are 8 backend access windows each FFT cycle. Each access provides 8 bytes. Thus the maximum capacity is 8 accesses * 8 bytes = 64 bytes per 16.125 usec or 3.97 MByte/sec. This bandwidth will be shared between the FILTER (FIR), the LTA 8751 microprocessor and the refresh logic. The FIR does not take advantage of every possible LTA backend access. The FIR skips one possible access out of every five. Thus the FIR will use at most 44,236 of the available backend accesses. This translates to 44,236 single precision complex results per 131 msec interval, or 353,888 bytes per 131 msec interval, or 2.7 MByte/sec. In terms of baselines, the 44,236 results per 131 msec interval translates to 21 baselines of 2048 results each or 43 baselines of 1024 results each.

The order of priority for the backend access during the first seven access windows of each FFT cycle is:

highest:	filter	
	refresh	(if no refresh yet in FFT cycle)
	8751	
lowest:	refresh	(if refresh has occurred in FFT cycle)

During the eighth access window of the FFT cycle, if refresh has not yet occurred, the highest priority will become refresh. This will insure that at least one refresh occurs per FFT cycle.

In order to (almost) meet the refresh requirement of 512 refreshes in 8 ms, one refresh would be required per FFT cycle. (512 * 16.125 usec = 8.256 msec)

The backend access window is also used for writing the mode word to the floating point chips. This has the highest priority, but only occurs at startup.

Where different groups have different accumulation times, it will be necessary to partition transfers. For example, if one group has an accumulation time of 131 msec, and another has an accumulation time of 13.1 seconds, all results for the first group must be transferred every 131 msec, along with some fraction of the results for the second group (in two different transfers).

The output interface on the LTA card is a 32 bit wide "read only" parallel output, consisting of the following lines:

32 bit output data bus

20 bit input address (1 bit bank field, 8 bit baseline field, 11 bit result field):

result field goes through a ram lookup translation which is a function of the mapping code associated with the baseline field

1 bit input control to force bank selection

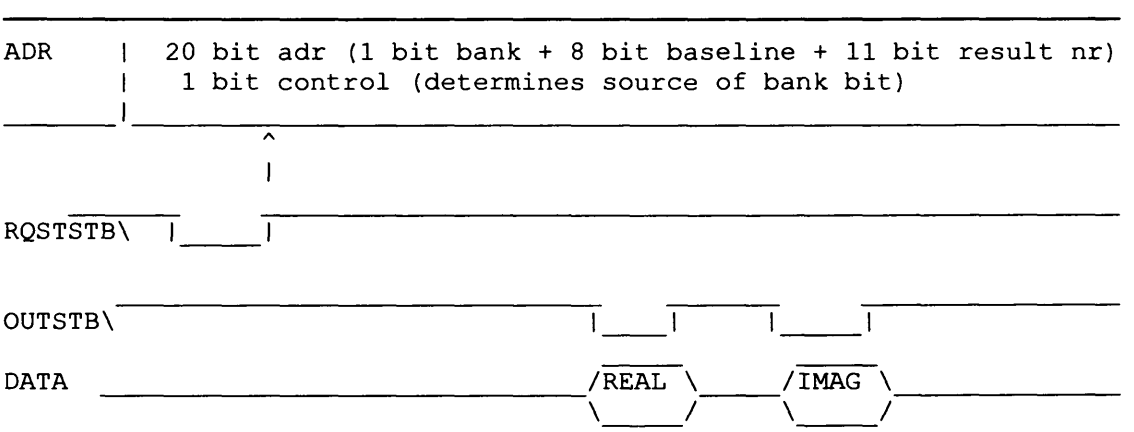
(disable automatic bank selection and use the bank bit in address)

1 bit input strobe to request a result (32 bits REAL and 32 bits IMAG)

1 bit output strobe to write the REAL 32 bits to the FIR input FIFO, followed by the IMAG 32 bits

The general sequence of events for reading a complex result is as follows:

Figure 10 LTA BACKEND ACCESS

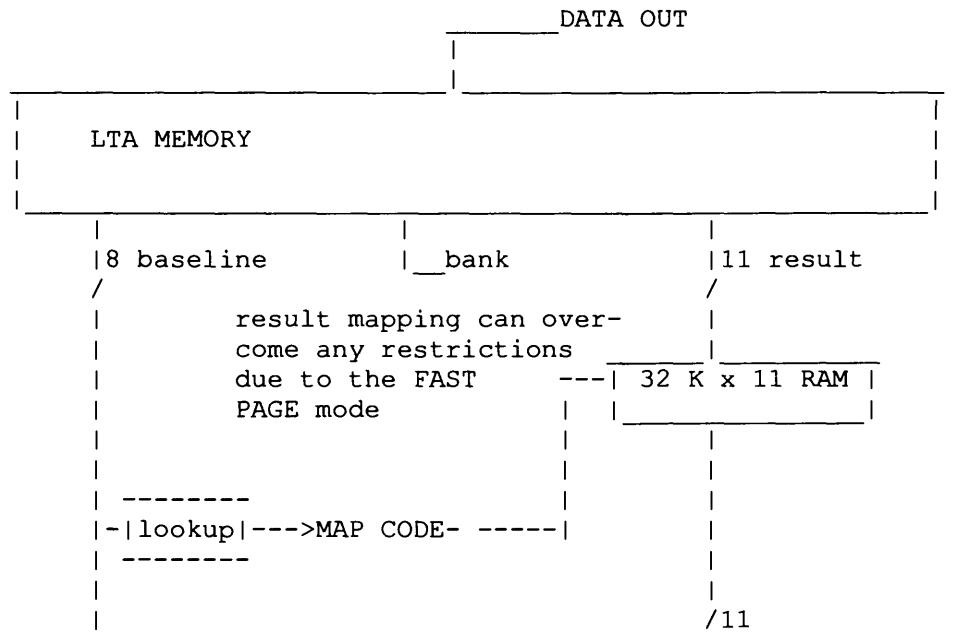


For reading validities, the validity value is duplicated in the REAL and IMAG fields. For all reads of the LTA DRAM, two transfers will occur, even if the full 64 bit result is not of interest.

If the RQSTSTB is properly synchronized to the FFT cycle, a result may be read 8 times per FFT cycle, at 2 uSec intervals.

The following figure indicates the general nature of the address decoding for backend accesses to the LTA:

Figure 11 LTA BACKEND ADDRESSING



Note regarding the result field translation:

If it were not for the feature of data reduction provided with the result field translation at the front end, then we could avoid translation at the front end and only do translation at the backend. If the DRAM fast page mode did not force the previously mentioned addressing restriction, then no translation would be needed at the backend. As it is, translation is provided at both ends. It is still possible that no translation will be required at the back end. See Section 4.2.7 for more details (the translation function has been moved to the FIR output).

The LTA memory is both readable and writable from the 8751 CPU. This is done by making the backend access available to the CPU whenever the FIR or refresh is not using it.

4.2.3 Validities

For transfer of validities from the MAC to the LTA, the following "artificial" baseline numbers and map codes are used as flags in the hardware (map number ls bit is used to distinguish tape from pulsar validity):

Baseline Number	Map Code	Function
210	8,10,12 or 14	Tape Data Validates bslines 0-209
211	9,11,13 or 15	Pulsar vals chs 0,2,4 and 6; 4 racks
212	9,11,13 or 15	Pulsar vals chs 1,3,5 and 7; 4 racks

Baseline tape data validities are stored along the spectral point axis of artificial "baseline" number 210. Each MACWE makes the contents of one Tape Validity counter available in each array 0-7; only the first 210 MACWEs out of the full sequence of 256 are of interest):

MACWE #	"Baseline" 210 STORAGE LOC	Array Number	Tape Validity for Hardware Baseline
0	0-7	0-7	0
1	8-15	0-7	1
2	16-23	0-7	2
|	|	|	|
|	|	|	|
209	1672-1679	0-7	209
210			
|	no results of interest for MACWEs #210 - #255		
255			

where the MACWE # is the access in a four cycle gap.

The normal selection of the bank switch bit is done by applying the accumulation code 0 thru 7 (from the baseline lookup ram) to a multiplexer to select the bank switch bit for the group. When "baseline" 210 is coming in, map code 8 (for example) comes from the lookup to indicate that the incoming value is a tape data validity. This causes a separate lookup to occur, to translate the MAC point number to an accumulation code for the particular hardware baseline and this accumulation code is used to select the proper LTA bank for the validity.

The ram to look up the accumulation code for tape validities thus needs to be at least 256 bytes deep. It is deeper in order to handle pulsar validities.

Pulsar validities are stored along the spectral point axis of artificial "baselines" number 211 and 212. There is one Master Control Card per rack. Each one of these four cards maintains 256 pulsar validity counters for each of four channels. This is 1024 pulsar validity counters per card, 4096 counters total.

The pulsar validity counters in rack 0 are for stations 0-9, channels 0-3. Rack 1 handles stations 10-19, channels 0-3. Rack 2 handles stations 0-9, channels 4-7. Rack 3 handles stations 10-19, channels 4-7.

We are required to support pulsar observations in only one group. NOTE: GENERAL SUPPORT OF MORE THAN ONE PULSAR GROUP HAS BEEN ELIMINATED TO ALLOW LTA LOOKUP TABLES TO HAVE BANKS FOR PRE-LOADING. More limited support of more than one pulsar group is still possible as discussed below.

The hardware will support two groups of separate pulsar observations, if the groups are divided such that only stations 0-9

are in one group and only stations 10-19 are in the second group. In this case, racks 0 and 2 handle one group while racks 1 and 3 handle the second group.

In a more restricted manner, the hardware will support four groups of separate pulsar observations as in the following example:

```
stations  0 -  4, ch 0-3 group #1
stations  5 -  9, ch 0-3 group #2
stations 10 - 14, ch 4-7 group #3
stations 15 - 19, ch 4-7 group #4
```

where four channels from each station are "wasted".

There is a further restriction on the use of more than one group in pulsar observations. The fast page mode restriction forces all pulsar groups to have the same dump rate, so that the bank bit is the same for all 8 pulsar validities associated with a single baseline and MACWE. (The set of 8 results represent four different groups, which must all go into the same half of the 2048 result field of the DRAM.)

The following tables list the storage locations in baselines 211 and 212 when no result translation is used (they show the order in which results are transferred into the LTA):

MACWE #	"Baseline" 211 LOC	Rack	Array Nr	Stations	Pulsar Validity Counter	3 LS BITS OF LOC aryA	aryB
0	0-1	0	0,2	0 - 9	0	000	001
	2-3	1	0,2	10 - 19	0	010	011
	4-5	2	4,6	0 - 9	0	100	101
	6-7	3	4,6	10 - 19	0	110	111
1	8-9	0	0,2	0 - 9	1	000	001
	10-11	1	0,2	10 - 19	1	010	011
	12-13	2	4,6	0 - 9	1	100	101
	14-15	3	4,6	10 - 19	1	110	111
2	16-17	0	0,2	0 - 9	2		
	18-19	1	0,2	10 - 19	2		
	20-21	2	4,6	0 - 9	2		
	22-23	3	4,6	10 - 19	2		
3	24-25	0	0,2	0 - 9	3		
	26-27	1	0,2	10 - 19	3		
	28-29	2	4,6	0 - 9	3		
	30-31	3	4,6	10 - 19	3		
254	2032-2033	0	0,2	0 - 9	254		
	2034-2035	1	0,2	10 - 19	254		
	2036-2037	2	4,6	0 - 9	254		
	2038-2039	3	4,6	10 - 19	254		
255	2040-2041	0	0,2	0 - 9	255		
	2042-2043	1	0,2	10 - 19	255		
	2044-2045	2	4,6	0 - 9	255		
	2046-2047	3	4,6	10 - 19	255		

bits 2^1 and 2^2 count the rack nr

MACWE #	"Baseline" 212 STORAGE LOC	Rack	Chan Nr	Stations	Pulsar Validity Counter
0	0-1	0	1,3	0 - 9	0
	2-3	1	1,3	10 - 19	0
	4-5	2	5,7	0 - 9	0
	6-7	3	5,7	10 - 19	0
1	8-9	0	1,3	0 - 9	1
	10-11	1	1,3	10 - 19	1
	12-13	2	5,7	0 - 9	1
	14-15	3	5,7	10 - 19	1
2	16-17	0	1,3	0 - 9	2
	18-19	1	1,3	10 - 19	2
	20-21	2	5,7	0 - 9	2
	22-23	3	5,7	10 - 19	2
3	24-25	0	1,3	0 - 9	3
	26-27	1	1,3	10 - 19	3
	28-29	2	5,7	0 - 9	3
	30-31	3	5,7	10 - 19	3
⋮	⋮	⋮	⋮	⋮	⋮
254	2032-2033	0	1,3	0 - 9	254
	2034-2035	1	1,3	10 - 19	254
	2036-2037	2	5,7	0 - 9	254
	2038-2039	3	5,7	10 - 19	254
255	2040-2041	0	1,3	0 - 9	255
	2042-2043	1	1,3	10 - 19	255
	2044-2045	2	5,7	0 - 9	255
	2046-2047	3	5,7	10 - 19	255

If we only support a single group doing pulsar observations, then the determination of the bank select bit involves placing a single accumulation code in every location of the portion of the lookup ram that deals with baselines 211 and 212. The ram address is then formed from the 8 bit result field plus the LS bit of the mapping code, and the ram depth is 512 bytes. The LS bit of the mapping code distinguishes between mapping code 8 for tape validities and mapping code 9 for pulsar validities.

The following descriptions of the lookup rams for validity bank selection (for both the front end and back end) are based on the results being stored in the DRAM baselines 210, 211 and 212 as detailed above. These sequences are in turn based on the front end result number translations being transparent ("bypassed") for validities.

For tape validities ("artificial" baseline 210), the accumulation code to be looked up is a function of the "real" baseline number which is

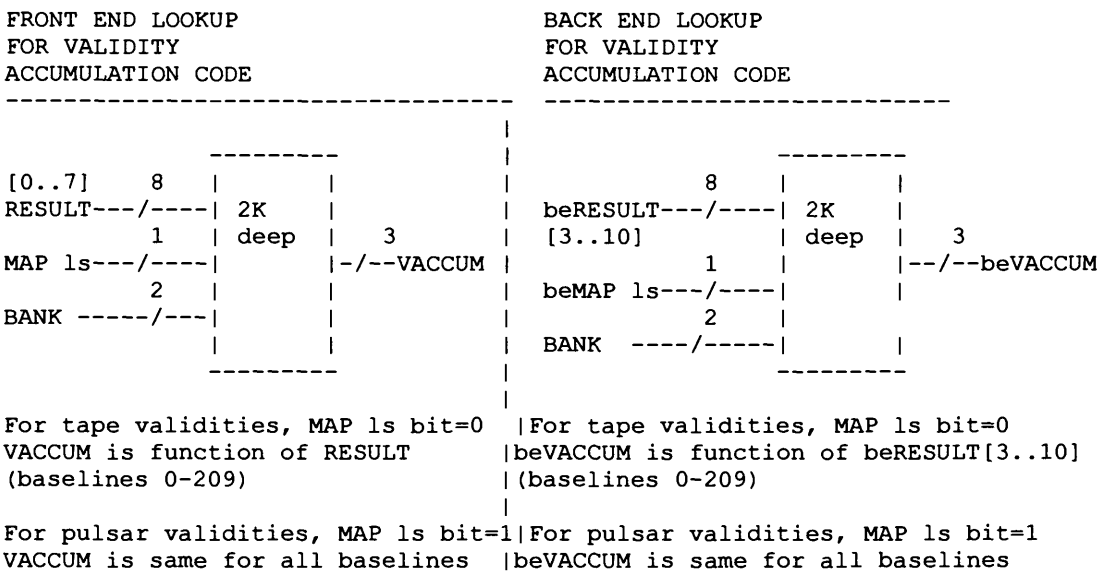
the same as the result number (result 0-209 = baseline 0-209).

For pulsar validities ("artificial" baselines 211 and 212), the accumulation code is the same for all baselines.

A similar lookup ram is required on the backend in order to automatically select the proper bank for backend access of validities. (It is possible for the backend to keep track of LTA bank switching independently and override the automatic selection.) This ram is 2K bytes deep. The address field is made up from the MS 10 bits of the 11 bit backend result field and the LS bit of the backend mapping code.

For "artificial" baseline 210 (tape validities), the mapping code number is even (LS bit = 0) and the lookup is a function of the MS 8 bits
of the 11 bit backend result field.

For "artificial" baselines 211 and 212 (pulsar validities), the mapping code is odd (LS bit = 1) and the lookup is the same for all baselines.



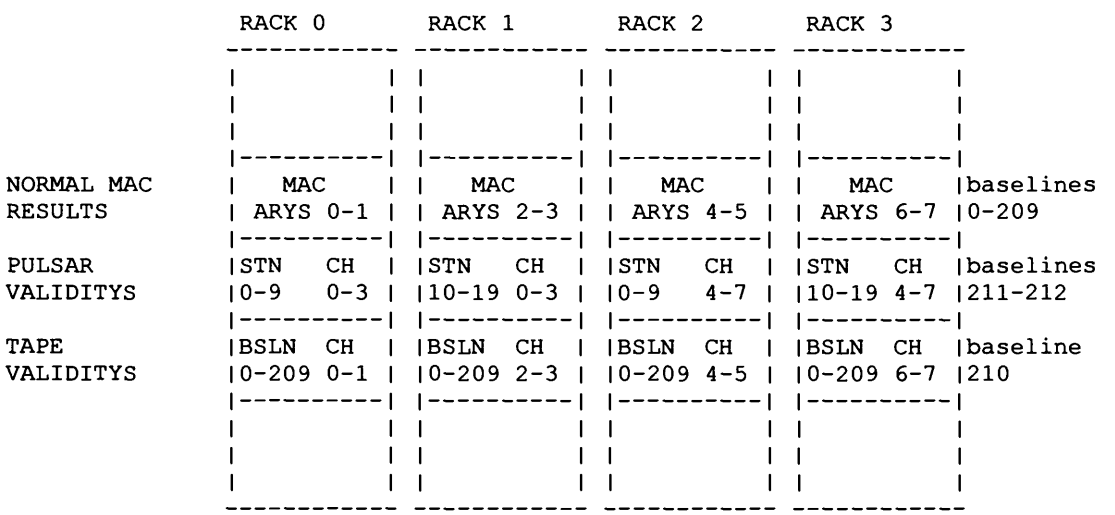
4.2.4 Details of sequence of results from MAC quadrants to LTA:

Each MACWE causes one result in every MAC VLBA1 to be strobed to secondary storage in the VLBA1. Thus there is one result in each of 8 arrays for each of 210 MAC chips ready to be read into the LTA. This is 8 * 210 = 1680 results. For each MACWE, we also transfer validity results to the LTA, using three "artificial" baselines. Baseline #210 is used for tape validities and baselines #211-212 are used for pulsar validities. Thus we count through baseline numbers 0-212 and transfer 8 results for each of these baselines for each of 256 MACWEs.

For normal results, the total is:
256 MACWEs * 210 baselines * 8 arrays = 430,080.

For tape validities, the total is:
256 MACWEs * 1 baseline * 8 results = 2048. (Only 1680 are real.)

For pulsar validities, the total is:
256 MACWEs * 2 baselines * 8 results = 4096.



For each MACWE, the following results are transferred from each of four racks to the LTA:

MACWE#	RACK 0			RACK 1			RACK 2			RACK 3			
	rslt	bsln	ary	rslt	bsln	ary	rslt	bsln	ary	rslt	bsln	ary	
0	0	0	0	0	0	2	0	0	4	0	0	6	
0	0	0	1	0	0	3	0	0	5	0	0	7	
0	0	1	0	0	1	2	0	1	4	0	1	6	
0	0	1	1	0	1	3	0	1	5	0	1	7	
													Normal results
0	0	2	0	0	2	2	0	2	4	0	2	6	point # 0
0	0	2	1	0	2	3	0	2	5	0	2	7	210 baselines
|	|	|	|	|	|	|	|	|	|	|	|	|	
|	|	|	|	|	|	|	|	|	|	|	|	|	
0	0	209	0	0	209	2	0	209	4	0	209	6	
0	0	209	1	0	209	3	0	209	5	0	209	7	
0	0	210	0	0	210	2	0	210	4	0	210	6	Tape Validity
0	0	210	1	0	210	3	0	210	5	0	210	7	baseline 0
0	0	211	0	0	211	0	0	211	4	0	211	4	Pulsar Validity
0	0	211	2	0	211	2	0	211	6	0	211	6	counter 0
0	0	212	1	0	212	1	0	212	5	0	212	5	Pulsar Validity
0	0	212	3	0	212	3	0	212	7	0	212	7	counter 0
1	1	0	0	1	0	2	1	0	4	1	0	6	
1	1	0	1	1	0	3	1	0	5	1	0	7	
1	1	1	0	1	1	2	1	1	4	1	1	6	
1	1	1	1	1	1	3	1	1	5	1	1	7	
													Normal results
1	1	2	0	1	2	2	1	2	4	1	2	6	point # 1
1	1	2	1	1	2	3	1	2	5	1	2	7	210 baselines
|	|	|	|	|	|	|	|	|	|	|	|	|	
|	|	|	|	|	|	|	|	|	|	|	|	|	
1	1	209	0	1	209	2	1	209	4	1	209	6	
1	1	209	1	1	209	3	1	209	5	1	209	7	
1	1	210	0	1	210	2	1	210	4	1	210	6	Tape Validity
1	1	210	1	1	210	3	1	210	5	1	210	7	baseline 1
1	1	211	0	1	211	0	1	211	4	1	211	4	Pulsar Validity
1	1	211	2	1	211	2	1	211	6	1	211	6	counter 1
1	1	212	1	1	212	1	1	212	5	1	212	5	Pulsar Validity
1	1	212	3	1	212	3	1	212	7	1	212	7	counter 1

MACWE#	RACK 0			RACK 1			RACK 2			RACK 3			
	rslt	bsln	ary	rslt	bsln	ary	rslt	bsln	ary	rslt	bsln	ary	
2	2	0	0	2	0	2	2	0	4	2	0	6	
2	2	0	1	2	0	3	2	0	5	2	0	7	
2	2	1	0	2	1	2	2	1	4	2	1	6	
2	2	1	1	2	1	3	2	1	5	2	1	7	
													Normal results
2	2	2	0	2	2	2	2	2	4	2	2	6	point # 2
2	2	2	1	2	2	3	2	2	5	2	2	7	210 baselines
|	|	|	|	|	|	|	|	|	|	|	|	|	
|	|	|	|	|	|	|	|	|	|	|	|	|	
2	2	209	0	2	209	2	2	209	4	2	209	6	
2	2	209	1	2	209	3	2	209	5	2	209	7	
2	2	210	0	2	210	2	2	210	4	2	210	6	Tape Validity
2	2	210	1	2	210	3	2	210	5	2	210	7	baseline 2
2	2	211	0	2	211	0	2	211	4	2	211	4	Pulsar Validity
2	2	211	2	2	211	2	2	211	6	2	211	6	counter 2
2	2	212	1	2	212	1	2	212	5	2	212	5	Pulsar Validity
2	2	212	3	2	212	3	2	212	7	2	212	7	counter 2

etc. up through MACWE # 255.

In all cases the result number matches the MACWE number.

For normal results, this is the address of one of the 256 results in the MAC VLBA1. In the case of 512 point FFTs, this is also the spectral channel point number. For other FFT sizes, this is not the case. For smaller FFT sizes, spectral channel point number 0 for all "adjacent" ffts come first, followed by spectral channel point number 1 etc. For larger FFT sizes, spectral channels are spread out between more than one array.

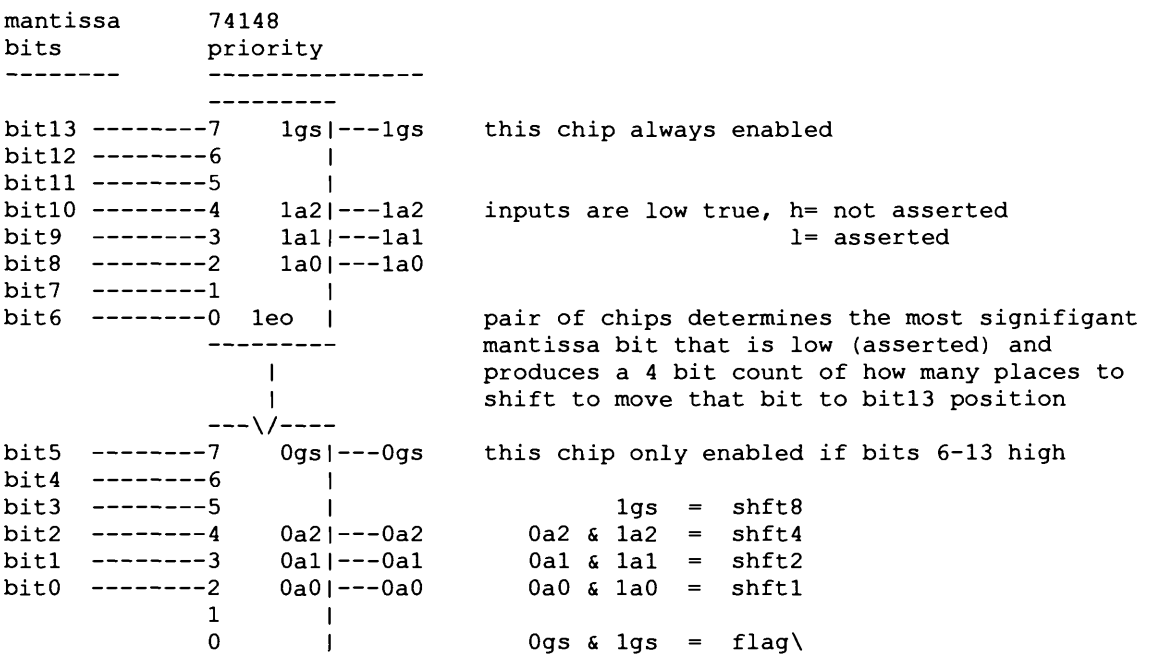
For tape validities, result numbers 0-209 correspond to actual baseline numbers 0-209, and result numbers 210-255 are meaningless.

For pulsar validities, result numbers 0-255 correspond to the validity counters 0-255 which are associated with the spectral point order of the normal results.

4.2.5 Details of 15,15,6 to IEEE conversion:

The following priority encoder is implemented in the Xilinx at location 20J: (see L024D11.XIL - L024D15.XIL also)

(The 15 bit mantissas are one's complement format, the 6 bit exponent is two's complement)



TRUTH TABLE FOR THE PRIORITY ENCODER ABOVE:

2	3	4	5	6	7	0	1	2	3	4	5	6	7										
0	1	2	3	4	5	6	7	8	9	10	11	12	13	1a2	1a1	1a0	1gs	1eo	0a2	0a1	0a0	0gs	
h	h	h	h	h	h	h	h	h	h	h	h	h	h	1	1	1	1	0	1	1	1	1	
x	x	x	x	x	x	x	x	x	x	x	x	x	l	0	0	0	0	1	1	1	1	1	
x	x	x	x	x	x	x	x	x	x	x	x	l	h	0	0	1	0	1	1	1	1	1	
x	x	x	x	x	x	x	x	x	x	x	l	h	h	0	1	0	0	1	1	1	1	1	
x	x	x	x	x	x	x	x	x	x	l	h	h	h	0	1	1	0	1	1	1	1	1	
x	x	x	x	x	x	x	x	x	l	h	h	h	h	1	0	0	0	1	1	1	1	1	
x	x	x	x	x	x	x	x	l	h	h	h	h	h	1	0	1	0	1	1	1	1	1	
x	x	x	x	x	x	x	l	h	h	h	h	h	h	1	1	0	0	1	1	1	1	1	
x	x	x	x	x	x	l	h	h	h	h	h	h	h	1	1	1	0	1	1	1	1	1	
x	x	x	x	x	l	h	h	h	h	h	h	h	h	1	1	1	1	0	0	0	0	0	
x	x	x	x	l	h	h	h	h	h	h	h	h	h	1	1	1	1	0	0	0	1	0	
x	x	x	l	h	h	h	h	h	h	h	h	h	h	1	1	1	1	0	0	1	0	0	
x	x	l	h	h	h	h	h	h	h	h	h	h	h	1	1	1	1	0	0	1	1	0	
x	l	h	h	h	h	h	h	h	h	h	h	h	h	1	1	1	1	0	1	0	0	0	
l	h	h	h	h	h	h	h	h	h	h	h	h	h	1	1	1	1	0	1	0	1	0	

From the truth table above, the shft values and flag as a function of the most significant asserted mantissa bit are:

bit	shft value	flag\
none	15	1
13	0	0
12	1	0
11	2	0
10	3	0
9	4	0
8	5	0
7	6	0
6	7	0
5	8	0
4	9	0
3	10	0
2	11	0
1	12	0
0	13	0

At the input to the shifter, the mantissa bits are shifted by one place in the wiring. This places bit 13 in the phantom bit position.

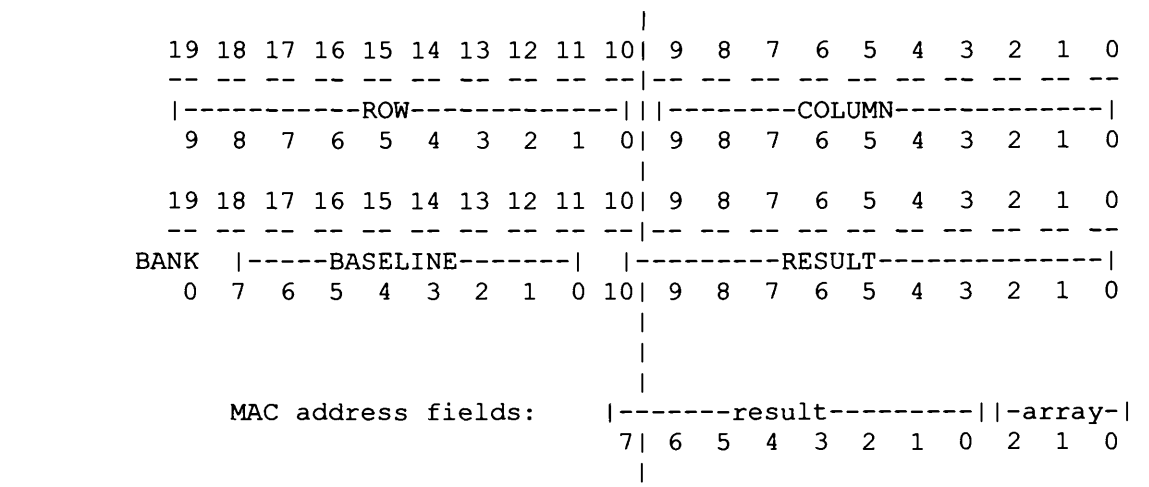
A prom lookup table determines the IEEE exponent field. To convert a 15,15,6 value to IEEE, the prom adds "excess 127" to the 6 bit exponent and then subtracts the number of places of shift and subtracts 1 more. The subtraction of 1 is due to the fact that the 15,15,6 value is fractional and the IEEE value is between 1.0 and 2.0. To convert a validity integer to IEEE, the prom adds "excess 127" plus an additional 13 to the 6 bit exponent and then subtracts the number of places of shift and subtracts 1 more.

If the value is zero, the flag\ bit is used by the PROM to set the IEEE exponent field to zero. (The mantissas will be shifted by 15 places, but all bits are zero before and after the shift.)

See LTA sheet L024D07.DOC for additional details.

The contents of the proms at locations 21G and 22G are generated by the C program d019ieee.c, maintained in vlbsoft/pal_prom/proms/ltaseq/SCCS. The program produces a binary file named d019ieee.bin which is read into a PROM programmer to program the two proms. Both proms are programmed with the same binary file, so they are identical.

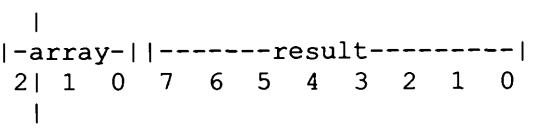
4.2.6 LTA input address mapping and Fast Page Mode:



The 11 bit RESULT field of the LTA DRAM address is derived from the 3 bit array field and 8 bit result field of the "address" of the MAC results. The address mapping on the LTA input would allow any arbitrary mapping from the 11 bit MAC address to the 11 bit LTA address as a function of the mapping code associated with the baseline. The Fast Page Mode of the DRAM imposes the restriction that the three bits of array in the MAC address must all be within the column field. This is due to the fact that results are transferred, using Fast Page Mode, from the MAC to the LTA in sets of 8, where the 8 results are one each from each array (the only bits changing in the MAC address are the 3 bits of the array field). Present plans call for no mapping to be done, except for data reduction which is discussed below.

4.2.7 LTA output address mapping:

A full 11 bit to 11 bit address translation is provided as a function of the mapping code associated with the baseline when reading the LTA results. The output access is fully random (does not use Fast Page Mode). Thus, for example, the 11 bits might be re-ordered in a more natural order as follows:



As design of the FIR (filter) evolved, the final address mapping was moved to the output of the FIR. This then allowed the three bit array field to stay on the right, so that even after data reduction (discussed below) the lower bits may be used to determine which tape validity applies to the result.

The output address mapping thus is not presently used in the LTA, but it will be left available for use.

8-22-91:
In conversation with Ray, it seems that validities are a bit more involved when data reduction is occurring. If arrays are being added together, Ray believes it is possible for the validity count for one of the arrays to be different from the other (assuming two arrays being added together). Thus the validities also need to added together in the LTA. This should be possible with the input translation table. When adjacent FFTs or spectral points are being added together, the validities for each are identical, thus there is a constant scale factor that can be applied to the results.

If validities are added together in the LTA, and if the FIR still needs 8 distinct validities to "blindly" apply, the LTA output result translation table can be used to provide multiple copies of the same validity to the FIR.

A single translation table for baseline 210 and a second table for baselines 211 and 212 can provide the necessary data reduction for validities.

4.2.7.1 Data Reduction:

When data reduction is being used, there are fewer results per baseline stored in the LTA than there are available from the MAC. The 11 bit result field thus becomes smaller. The three bit array field in the MAC address that will be used to do a look up to see which tape validity applies to each array will thus change width (become 2, 1 or zero bits wide). Data reduction will be done in such a manner that the array field remains on the right, and both fields "compress" to the right.

```
256 point FFT, 8 chan mode          |
128 results                         |-------result-------|-array--|
each of 8 arrays                   x| 6  5  4  3  2  1  0  2  1  0
                                    |

64 point FFT, 8 chan mode           |
32 results                          |      |--result-----|-array--|
each of 8 arrays                   x| x  x  4  3  2  1  0  2  1  0
                                    |

64 point FFT, 4 chan mode           |
32 results                          |        |--result------|-ary-|
each of 4 pairs of arrays          x| x  x  x  4  3  2  1  0  1  0
(pairs of arrays added together)    |
```

4.2.7.2 Clear Accumulator:

When one or more data reductions (spectral averaging, transform averaging or array averaging) is occurring, the CLAC Inhibit bit from the FERSLT table must be used to allow CLAC to occur for the first result to be accumulated into a single location but inhibited for all other results that also accumulate into the same original location.

Thus, where the FERSLT table contains the LTA result number into which each MAC result will accumulate, it must also have the MS bit cleared for the first MAC result to go into a specific location, and have the MS bit set for each subsequent result that is accumulated into the same original location.

The following table provides a simple example for spectral averaging of 4 spectral channels into one:

Array	MAC Result	LTA Result	CLAC INH	
0	0	0	0	the 256 MAC results
0	1	0	1	are reduced to 64 LTA
0	2	0	1	results;
0	3	0	1	
0	4	8	0	with no array averaging,
0	5	8	1	LTA results are stored
0	6	8	1	with the array field
0	7	8	1	changing most rapidly
0	8	16	0	thus the LTA result
0	9	16	1	column varies in steps
0	10	16	1	of 8 as shown
0	11	16	1	
0	12	24	0	
0	13	24	1	we allow CLAC to be asserted
0	14	24	1	for the first spectral result
0	15	24	1	of each set of four, but
				then inhibit CLAC for the
				remaining three results since
1	0	1	0	they accumulate into the same
1	1	1	1	LTA result that the first of
1	2	1	1	the set went into
1	3	1	1	
1	4	9	0	
1	5	9	1	
1	6	9	1	
1	7	9	1	
7	0	7	0	
7	1	7	1	
7	2	7	1	
7	3	7	1	
7	248	510	0	
7	249	510	1	
7	250	510	1	

7	251	510	1
7	252	511	0
7	253	511	1
7	254	511	1
7	255	511	1

If we now assume a 256 point FFT is being done so that we are doing transform averaging:

Array	MAC Result	LTA Result	CLAC INH	
0	0	0	0	here we allow CLAC to
0	1	8	1	be asserted every other
0	2	16	0	time, because spectral
	|	|		point 0 from the first
	|	|		transform comes out first
0	126	1008	0	followed by spectral
0	127	1016	1	point 0 from the second
0	128	0	0	transform, etc.
0	129	8	1	
0	130	16	0	
	|	|		
	|	|		
0	254	1008	0	
0	255	1016	1	

If we now assume adjacent Arrays are being averaged in pairs:

Array	MAC Result	LTA Result	CLAC INH	
0	0	0	0	here we allow CLAC to
0	1	4	0	be asserted for the first
0	2	8	0	array, and then inhibit
0	3	16	0	CLAC for the second array
	|	|		that is summed into the
	|	|		same LTA locations as the
1	0	0	1	first array
1	1	4	1	
1	2	8	1	
1	3	16	1	

4.2.8 Detailed Description of Pulsar Validities:

Pulsar validities are Short Term Accumulated (STA) on the Master Control Cards (MCC). There is one MCC per rack. Each one of these four cards maintains four sets of 256 pulsar validity counters, one for each of the four FFT pipelines, for one set of ten stations (FFT 0-9 or 10-19). This produces 1024 pulsar validity counters per MCC and 4096 counters total for the VLBA Correlator. The 16 sets of 256 counters, are accumulated in LTA baselines 211 and 212 (2048 validity counts in each of the two baselines) as follows:

MCC	FFT STATIONS	FFT PIPELINE	LTA BASELINE
0	0-9	0	211
	0-9	2	211
	0-9	1	212
	0-9	3	212
1	10-19	0	211
	10-19	2	211
	10-19	1	212
	10-19	3	212
2	0-9	4	211
	0-9	6	211
	0-9	5	212
	0-9	7	212
3	10-19	4	211
	10-19	6	211
	10-19	5	212
	10-19	7	212

The 256 pulsar validity counters for one FFT pipeline contain the counts of how many FFT cycles each spectral point from the associated FFT pipeline was valid during a 131 msec. STA cycle. The order of the validity counters is the same as the order of transfer of visibilities from FFT to MAC. That is to say, for a 512 point FFT with 256 spectral channel results, the results are transferred from FFT to MAC in spectral order (0-255), and pulsar validity counter #0 represents visibility spectral point #0, pulsar validity counter #1 represents visibility spectral point #1 etc. For a 256 point FFT with 128 spectral results, there are two "adjacent" 256 point transforms in the 512 point FFT pipeline. The results are transferred in the order of both spectral points 0 first followed by both spectral points 1 etc. Thus, pulsar counter #0 represents spectral channel #0 of the first transform in the pipeline and pulsar counter #1 represents spectral channel #0 of the second transform. The pulsar validity counters are counting the actual pulsar gate. The gate must be generated based on a knowledge of the order of results out of the FFT pipeline. The meaning of each pulsar validity counter is thus

directly related to the address sequence of results out of the FFT pipeline.

As pulsar validities are transferred from the MCC to the LTA, they come in groups of 8, similar to the visibilities, and must also all be mapped into the same page of DRAM due to the Fast Page Mode operation. This means that the contents of the result mapping table for the pulsar validities, being an 11 bit value capable of representing 0-2047, must hold the MS bit unchanged for each group of 8 results being mapped. Even though some limited "sub-arraying" in pulsar mode may be possible, the fast page mode restriction requires that all such pulsar observations use the same accumulation code, so that the eight results in each group always go into the same bank, since the BANK bit is the MS bit of the 20 bit address to the DRAM, and the upper 10 bits of the address cannot change during Fast Page Mode accesses.

A DRAM page is 1024 locations, while a baseline contains 2048 results. Thus if no averaging is done going into the LTA, the Fast Page Mode restriction is of concern. If averaging is being done, then there are 1024 or less of the 2048 locations in a LTA baseline being addressed and the Fast Page Mode restriction becomes moot for visibilities.

The 20 bit address to the ram is as follows:

```
19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
|-----------ROW-------------| |--------COLUMN-------------|
 9  8  7  6  5  4  3  2  1  0  9  8  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
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
BANK  |-----BASELINE-------|  |---------RESULT--------------|
 0  7  6  5  4  3  2  1  0 10  9  8  7  6  5  4  3  2  1  0
 |           |                          |
 |__2 banks  |___256 baselines          |___2048 results
               (210 + 46 "spare")
```

Results are transferred from the STA to the LTA in 256 blocks associated with the MACWE (MAC Write Enable) accesses to the STA. Every 131 msec., the 256 spectral results available in each VLBA1 asic are transferred from asic ram to asic output register one at a time. The MACWE signal is globally asserted to every asic (210 asics in each of 8 arrays) along with a ram address. MACWE #0 accesses ram address 0 which is spectral result 0. MACWE #1 accesses ram address 1 etc., through MACWE #255 which accesses ram address 255. These ram addresses are in the "off line" bank in the asic.

For MACWE access #0, spectral result 0 is made available for readout from each of the 210 asics (210 baselines) in each of the 8 arrays. These are for baselines 0-209. Additionally, tape validity #0 is made available for baseline 210 from each of the 8 arrays, and pulsar counter #0 is made available from each of the 16 pulsar counter sets for baselines 211 and 212 (8 in each of the two baselines).

The table on the next page attempts to define the results transferred from STA to LTA each MACWE. For each of the 256 MACWE accesses, 1680 visibilities are transferred in 210 groups of 8 (these are the groups of 8 that use Fast Page Mode), along with 8 tape validities and 16 pulsar validities. The tape validities contain actual results only for the first 210 MACWE.

	1680 vis per MACWE		8 tape valids per MACWE		8 of 16 pulsar valids per MACWE			the other 8 of 16 pulsar valids per MACWE		
	BSLNS 0-209		BASELINE 210 rslt# =bsln#		BASELINE 211 rslt# =pulsar cntr#			BASELINE 212 rslt# =pulsar cntr#		
MAC WE	RE SLT	ARY	RE SLT	ARY	RE SLT	PIPE LINE	REPRESENTS	RE SLT	PIPE LINE	REPRESENTS
0	0	0	0	0	0	0	stn 0- 9	0	1	stn 0- 9
		1		1		2	stn 0- 9		3	stn 0- 9
		2		2		0	stn 10-19		1	stn 10-19
		3		3		2	stn 10-19		3	stn 10-19
		4		4		4	stn 0- 9		5	stn 0- 9
		5		5		6	stn 0- 9		7	stn 0- 9
		6		6		4	stn 10-19		5	stn 10-19
		7		7		6	stn 10-19		7	stn 10-19
1	1	0	1	0	1	0	stn 0- 9	1	1	stn 0- 9
		1		1		2	stn 0- 9		3	stn 0- 9
		2		2		0	stn 10-19		1	stn 10-19
		3		3		2	stn 10-19		3	stn 10-19
		4		4		4	stn 0- 9		5	stn 0- 9
		5		5		6	stn 0- 9		7	stn 0- 9
		6		6		4	stn 10-19		5	stn 10-19
		7		7		6	stn 10-19		7	stn 10-19
2	2	0	2	0	2	0	stn 0- 9	2	1	stn 0- 9
		1		1		2	stn 0- 9		3	stn 0- 9
		2		2		0	stn 10-19		1	stn 10-19
		3		3		2	stn 10-19		3	stn 10-19
		4		4		4	stn 0- 9		5	stn 0- 9
		5		5		6	stn 0- 9		7	stn 0- 9
		6		6		4	stn 10-19		5	stn 10-19
		7		7		6	stn 10-19		7	stn 10-19
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
209	209	0	209	0	209	0	stn 0- 9	209	1	stn 0- 9
		1		1		2	stn 0- 9		3	stn 0- 9
		2		2		0	stn 10-19		1	stn 10-19
		3		3		2	stn 10-19		3	stn 10-19
		4		4		4	stn 0- 9		5	stn 0- 9
		5		5		6	stn 0- 9		7	stn 0- 9
		6		6		4	stn 10-19		5	stn 10-19
		7		7		6	stn 10-19		7	stn 10-19
210	210	0	xxx	0	210	0	stn 0- 9	210	1	stn 0- 9
		1		1		2	stn 0- 9		3	stn 0- 9
		2		2		0	stn 10-19		1	stn 10-19
		3		3		2	stn 10-19		3	stn 10-19
		4		4		4	stn 0- 9		5	stn 0- 9
		5		5		6	stn 0- 9		7	stn 0- 9
		6		6		4	stn 10-19		5	stn 10-19
		7		7		6	stn 10-19		7	stn 10-19
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
255	255	0	xxx	0	255	0	stn 0- 9	255	1	stn 0- 9
		1		1		2	stn 0- 9		3	stn 0- 9
		2		2		0	stn 10-19		1	stn 10-19
		3		3		2	stn 10-19		3	stn 10-19
		4		4		4	stn 0- 9		5	stn 0- 9
		5		5		6	stn 0- 9		7	stn 0- 9
		6		6		4	stn 10-19		5	stn 10-19
		7		7		6	stn 10-19		7	stn 10-19

Tape and Pulsar validities may be re-mapped or averaged using the front end mapping tables of the LTA. Mapping tables 8, 10, 12 and 14 are available for tape validities. Tables 9, 11, 13 and 15 are available for pulsar validities.

Let's look at an example:

512 point FFT, 8 channel, spectral average by 2

	Visibilities				Pulsar Validities			
	Sequence of stored results NO averaging		Sequence of stored results averaged by 2		Sequence of stored results, baseline 211 same averaging by 2		in bsln 212	
LTA result	spec chan	array	spec chans	array	pulsar counter	pulsar set stn	pulsar set pipe	pulsar set stn	pulsar set pipe
0	0	0	0 + 1	0	0 + 1	0-9	0	0-9	1
1	0	1	0 + 1	1	0 + 1	0-9	2	0-9	3
2	0	2	0 + 1	2	0 + 1	10-19	0	10-19	1
3	0	3	0 + 1	3	0 + 1	10-19	2	10-19	3
4	0	4	0 + 1	4	0 + 1	0-9	4	0-9	5
5	0	5	0 + 1	5	0 + 1	0-9	6	0-9	7
6	0	6	0 + 1	6	0 + 1	10-19	4	10-19	5
7	0	7	0 + 1	7	0 + 1	10-19	6	10-19	7
8	1	0	2 + 3	0					
9	1	1	2 + 3	1					
10	1	2	2 + 3	2					
11	1	3	2 + 3	3					
12	1	4	2 + 3	4					
13	1	5	2 + 3	5					
14	1	6	2 + 3	6					
15	1	7	2 + 3	7					
|	|	|	|	|				
|	|	|	|	|				
|	|	|	|	|				
1016	127	0	254+255	0					
1017	127	1	254+255	1					
1018	127	2	254+255	2					
1019	127	3	254+255	3					
1020	127	4	254+255	4					
1021	127	5	254+255	5					
1022	127	6	254+255	6					
1023	127	7	254+255	7					
|	|	|	unused					
|	|	|	|	|				
|	|	|	|	|				
2040	255	0	unused						
2041	255	1							
2042	255	2							
2043	255	3							
2044	255	4							
2045	255	5							
2046	255	6							
2047	255	7							

The same result mapping table that is used for visibilities can be used for pulsar validities, but must be copied to map number 9, 11, 13 or 15.

The resulting sequence of storage along the 2048 point result axis of a baseline is the same except that for visibilities the items are in order by array (0-7) while pulsar validities are in order by "pulsar set"

The HCB protocol used to download the mapping table is shown below, using specific details related to pulsar validities.

HCB PROTOCOL AMAL ADDRESS

MS FIELD	MIDDLE	LS FIELD
adr bits [14..11]	adr bits [10..8]	adr bits [7..0]

MAP NR	PULSAR SET	PULSAR COUNTER	Data Bytes Transferred	
			MS Byte	LS Byte
0-15	0-7	0-255		
xxxx	yyy	zzzzzzzz	mmmmmmmm	lllllllll
			RR = 03	RR = 02

Example shown for Front End Result Mapping Table For Pulsar Validities

xxxx = 9, 11, 13 or 15

For each pulsar set 0-7 (what is represented by sets 0 - 7 depends on if this map is for baseline 211 or 212), provide a 256 entry list where data bytes MS,LS specify at which result number (0-2047) to store pulsar counter number Z, in pulsar set number Y.

20 OH OL FF BB PH PL RR AM AL CM CL DD----DD

OH OL specify the offset into the checksum table (OH = MS OL = LS)
FF MS nibble same as RR)below), LS nibble = 1
BB block number, dependent on RR, either 0-127, 0-3 or 0-7 for diff RR
PH PL checksum over the data block
RR specifies the lookup table ram (0-8)
AM AL specifies the start address (AM = MS AL = LS)
CM CL specifies the number of bytes (CM = MS CL = LS)

Transfers always begin at modulo 256 addresses, so that AL always = 0.
Transfers always contain 256 bytes, so CM = 1, CL = 0.
For the Front End Result Mapping table, RR = 02 transfers the LS bytes, RR = 03 transfers the MS bytes.

For pulsar validities, MAP NR must be 9, 11, 13 or 15. Pulsar sets are handled in the same order as arrays 0-7 and represent the following sets of pulsar validity counters:

For LTA Baseline 211:

Pulsar Set	Represents
0	Stations 0- 9, FFT pipeline 0
1	Stations 0- 9, FFT pipeline 2
2	Stations 10-19, FFT pipeline 0
3	Stations 10-19, FFT pipeline 2
4	Stations 0- 9, FFT pipeline 4
5	Stations 0- 9, FFT pipeline 6
6	Stations 10-19, FFT pipeline 4
7	Stations 10-19, FFT pipeline 6

For LTA Baseline 212:

Pulsar Set	Represents
0	Stations 0- 9, FFT pipeline 1
1	Stations 0- 9, FFT pipeline 3
2	Stations 10-19, FFT pipeline 1
3	Stations 10-19, FFT pipeline 3
4	Stations 0- 9, FFT pipeline 5
5	Stations 0- 9, FFT pipeline 7
6	Stations 10-19, FFT pipeline 5
7	Stations 10-19, FFT pipeline 7

4.3 LTA Circuit Description

Refer to schematic drawings L024D01.SCH through L024D04.SCH for the following circuit descriptions. The complete set of LTA drawings are numbered L024D01.SCH- L024D15.XIL, where sheets 1-4 are the schematic, sheets 5-10 are various block diagram and timing sequence type documentation and sheets 11-15 are Xilinx documentation. Drawings Z015D01.LAY through Z015D06.LAY are the IC layout and other layout type drawings (for various adapters etc.).

4.3.1 Input Circuit

The inputs from the MAC cards are by way of differential RS-485 signals. The bank of 75173 RS-485 to TTL receiver chips are shown on sheet 1 of the schematic. The output of these receivers are clocked into registers in the bank of PALCE22V10 PALs shown to the right of the receivers. These PALs are normally used as simple input registers, but when the TSTMODE input signal to the PALs is asserted, the PALs turn into pseudo random data generators to supply test patterns to the LTA in place of the MAC data.

4.3.2 15,15,6 To IEEE Conversion

The 15,15,6 format data from the MAC cards is transferred in two 18 bit fields. The conversion from 18 bit to 36 bit is handled in the input FIFOs shown to the right of the input registers.

Data from the output of the FIFOs is passed to the 15,15,6 to IEEE conversion stage found on sheet 3. The Xilinx XC3030 chip at 20J implements the required shifting functions. The PROMs at 21G and 22G provide the necessary exponent calculations. The 36 bit input value is converted into two single precision 32 bit IEEE values and passed to the AMD29C327 Floating Point Math Chips. There is one chip for the IEEE real value and one for the imaginary value. These chips perform

only the simple IEEE addition operation. The data sheet for the chip is included in Appendix VI of VLBA Technical Report No. 44.

The Xilinx personality was developed on a PC using the XACT design software, version 4.3 and the automatic place and route tool, APR, version 3.3. These tools require a hardware key. The Xilinx software tools are stored in and executed from the directory /home/azalea/pcstuff/xact.

The LTA Xilinx design files are maintained in the vlbsoft/xilinx/lta/SCCS directory. The top level macro file, d018lta1.mac, contains instructions for reproducing the design and the file mkltamcs.DOC contains additional general comments. The file d018lt01.mcs is the final deliverable product that the Makefile copies to the delivery area.

4.3.3 Sequencers

The operations of receiving, converting and accumulating the MAC data are controlled by the FFT Cycle Sequencer and the Fringe Cycle / Integration Cycle Sequencers shown on sheet 2 of the schematic. Timing sequences are shown on L024D08.DOC through L024D10.DOC. Sheet 8, in the lower right hand corner, shows the basic sequence for transferring a set of 8 results into the LTA from the 8 MAC arrays. The 15 bit real mantissa plus 3 bits of the exponent are transferred from arrays 0, 2, 4 and 6 followed by the 15 bit imaginary mantissa plus the other 3 bits of exponent from the same even numbered arrays. This is followed by similar transfers from the odd numbered arrays. The rest of sheet 8 shows the detailed timing sequence for processing the set of 8 results. As shown in the lower left corner of sheet 8, 8 such sets of 8 results (8 baselines total) are processed in a single FFT cycle.

L024D09.DOC presents the overall sequence for a complete 131 msec interval. L024D10.DOC presents the details of the sequencer output signals in the interval near the transition from the end of one 131 msec interval and the start of the next 131 msec interval.

The proms at locations 13D, 14D, 15E, 16D, 17D and 18D contain the 48 bit wide FFT cycle sequencer bits. The contents of these proms are generated by processing source files to produce binary image files. The source files are plain text files, one for each of the above prom chips. Each file has 8 columns defining the state of each bit in the prom at every address in the prom. The source file names are:

d019t13d.src, d019t14d.src, d019t15e.src,
d019t16d.src, d019t17d.src, d019t18d.src

and are processed by the C program named d019.c. These files are all maintained in the vlbsoft/pal_prom/proms/ltaseq/SCCS area. The program d019 reads an input file and writes an output file in binary format that is used by a prom programmer to program the chip. For example:

invoking the program as shown: d019 7b

will cause the file d019t7b.src to be opened and processed and the file d019t7b.bin will be written to disk. Invoking the program on each of the six source files will produce the six binary output files.

4.3.4 Front End Address Mapping

As results 0-255 from baselines 0-209 of arrays 0-7 are transferred into the LTA, there is an address lookup table that determines where each result will be accumulated into LTA DRAM. This table consist of the Front End Result Table (13H and 14H on sheet 3) and the Front End Baseline Table (13E on sheet 3). The baseline table allows for re-mapping the 210 hardware baselines into 210 LTA baselines. Presently this mapping is one-to-one. The result table allows for re-mapping the 256 results from each of 8 arrays into the 2048 results in each LTA baseline. Multiple mapping tables are provided for. Each hardware baseline is assigned to a result mapping table by way of the lookup table at 14E on sheet 3.

4.3.5 Bank Selection

The DRAM is double banked so that one bank can be readout while another is being used for accumulation. This bank assignment is done on a per baseline basis, using the most significant DRAM address bit (bit 19). Selection of the current bank for input is handled by the BANK REGISTER (3F on sheet 3) and the ACCUMULATION CODE derived from the lookup table in 14E on sheet 3. The contents of the BANK REGISTER are maintained by the 87C51 microprocessor, on 131 msec boundaries.

Bank selection for back end accesses are determined by the logic at 11E, 9D and 11D on sheet 3. For normal back end accesses, we read from the bank that is not being accumulated into, so the inverting output from 11D is used.

There is an un-used provision to allow the back end to force the bank for access by way of the BACK END FORCE bit seen as an input to 11D. This input is presently wired to logic 0 on the back plane at DIN pin 1A27. If the built in self test mode is ever implemented and if the self test results are to be transferred by way of the FIR instead of the HCB, then some work will be required in this area to make use of the BACK END FORCE signal. This is because at present, self test results are forced into bank 0 at all times but there is no provision to force the output to come from bank 0.

4.3.6 Clear Accumulator

The CLAC INHIBIT (CLAC) signal seen at 14H pin 19 on sheet 3 was discussed earlier. This signal from the front end result lookup table is required when data reduction is occurring so that more than one MAC result is accumulated into a single LTA result location. Normally CLAC is asserted for the duration of a 131 msec interval, when switching to a new bank. But when more than one MAC result is mapped to the same location, then CLAC must be asserted for the first result but not for

successive results. The PAL at 11G gates the CLAC INHIBIT signal with the CLAC signal from register 4F to provide this control.

4.3.7 Row and Column Multiplexing

The multiplexers in the lower center of sheet 3 provide the address multiplexing required to control accesses to the DRAM. The FIFOs at 17F and 16E provide a delay for the column address since each set of 8 results must be re-written to DRAM after having been read and accumulated.

4.3.8 87C51 Access

The 87C51 microprocessor is able to both read and write the DRAM as well as each of the lookup table rams. Data and address paths from the 87C51 to the various rams are seen in the lower left corner of sheet 2.

4.3.9 HCB Interface

A standard HCB interface is provided and seen on sheet 2. The 87C51 and Xilinx chip are downloaded over the HCB. All lookup table rams are also downloaded over the HCB. In addition, the DRAM results may be read out over the HCB. Before the FIR existed, this was the path by which correlator results were accessed.

4.3.10 Backend Interface

The Backend Interface is seen on sheet 1. Lookup tables similar to those on the front end are used, to allow re-mapping as required. The REQUEST STROBE signal from the FIR comes in on DIN pin 1A15 (RQSTSTB\) to request the 64 bit result for a given baseline and result number (presented to the chips at 22C, 22B and 21B). The requested result is read from the DRAM and written to the FIR by way of the FIR STROBE (FIRSTB) on DIN pin 1A17 and the 32 bit data bus from 74LAS374 registers (27D, 28D etc.).

4.4 LTA Utility Displays and Tests

There are no utility displays of much interest during normal observing. There are methods of accessing LTA results while observing. When testing, there are tests that can be run from the real time system and from the LTA serial port terminal.

4.4.1 Looking at LTA Results While Observing or Testing

The low level function rdlta() may be used to read raw results from the LTA. The parameter list for this function is:

```
rdlta(first_bsline, nr_bslines, first_result, nr_results)
```

when invoked at the vxWorks prompt. (The actual parameter list is longer, see hcbLtaTestFix.c for the complete function.) This function displays the real and imaginary values (real on left, imaginary on right) in floating point format and in four hex bytes per value (the raw IEEE format). This function is useful for displaying a few results from one or a few baselines. The total number of bytes requested must fit in a single HCB transfer. As an example,

```
rdlta 210,1,0,8
```

displays the validity counts for visibility baseline 0.

```
rdlta 210,1,8*209,8
```

displays the validity counts for visibility baseline 209. (Recall that the first 8 results at special baseline 210 are the validity counts for visibility baseline 0, the next 8 are for visibility baseline 1 etc.).

Complete spectra for any baseline may be readout and plotted. The following functions, found in hcbLtaTestFix.c may be used:

```
ltadump_avg (stnx, stny, chan, subarray, nr_chan_avg, fftsize, filename)
ltadump1K   (stnx, stny, chan, subarray, filename)
ltadump2K   (stnx, stny, chan, subarray, filename)
```

The function ltadump_avg() is used for FFT sizes 512 and smaller. When no spectral channel averaging is being used, nr_chan_avg equals 1. In all cases, a data set is written to the specified filename. The results in the file must be processed to format the data for plotting. A series of crude "back end processors" were written over the years for processing these LTA data dumps. The C source for these processors has been checked into the vlbsoft/ltaasm/SCCS directory. The executables are presently available in the directory /home/w5uxh/cbroadwe/ltabe. The present back end processor normally used is named ltabe5:

```
ltabe5 infilename, outfilename, nrtics, fftsize
```

The input file is the one written by the ltadump function. The number of tics is used in some statistical calculation that is an artifact. It is not necessary that the specified value matches the number of tics that was actually in use in the LTA. A value of 1 may be used as a default, assumming the statistical calculation is of no interest. The fftsize must be 64, 128, 256, 512, 1024 or 2048, and must agree with the ltadump function. The output file will be a text file suitable for using as input to the XMGR plotting utility on the SUN workstations. In a typical use of these functions, assuming an 8 channel, 512 point FFT observation:

```
ltadump_avg 4, 5, 0, 0, 1, 512, "test.dump"
```

will write results for station 4 times station 5, channel 0 (subarray = 0 specifies that the bank selection is based on accumulation code number 0). Self spectra for both stations 4 and 5 will be written to the file as well as the cross spectra. The self spectra are used for normalizing the cross spectra.

```
ltabe5 test.dump  test.gr  1 512
```

will read the input file and write the file test.gr which can be plotted by the xmgr program.

When dumping data from the LTA in the testbed rack, since there is not a complete MAC system available, the self spectra are not available. A modified version of the back end processor, ltabetb.c is used in these cases. Also, 1K and 2K FFTs present non-standard situations in the testbed. A brief example of processing a 2K cross spectra in the test bed is shown in W024D04.SCH, maintained in the corrdwgs/testbed/SCCS directory.

4.4.2 Real Time System Tests

The LTA may be tested in a stand alone fashion in the system or in the bench top test fixture or in the testbed rack. The standard test uses the pseudo random generator built into the input register PALs seen on sheet 1. When operating in the system or the testbed, where the LTA 131 msec is synchronized to the real time system 131 msec interrupt, the test is typically invoked as follows:

```
sp ltatest,nr_blocks
```

where nr_blocks specifies how much of the LTA storage to test. A parameter of 256 would test all results of all baselines. A full test takes a long time, so usually a smaller test is done. A typical value of nr_blocks is 5. Each block represents 8 results from each of 216 baselines in the LTA (baselines 0-215).

The test will specify a random number of 131 msec tics for accumulation, then after the accumulation interval is complete the specified number of blocks of results will be read and checked against predicted values. A new random accumulation time will then be selected and the test repeated. To stop the test, set the global variable ltatestflag = 0. The test will stop after the final check of the specified number of blocks of results.

When operating in the bench top test fixture without a 131 msec tic, an alternate test is available, invoked as follows:

```
sp ltatest1,nr_blocks
```

which operates in a similar manner, but does not require synchronization with a 131 msec tic between the LTA and the real time system.

These tests seem to fail the first time after power on initialization for some reason. This has not been investigated.

When the above tests are running, a status line indicates progress. If errors are detected in the accumulated results, the errors are reported on the screen.

4.4.3 Terminal Tests

From the terminal, a test similar to ltatest() can also be run. You can specify how many baselines to test and which of the 8 byte wide drams to test. This test also seems to fail the first time after power on initialization.

The parameters for the test are passed using the tf 0f test (one of the ram based built in test commands). Three parameters need to be passed, the first baseline to test, the last plus one to test and the mask to specify which dram bytes to test. A typical short test would specify baseline 0 through 4 and all eight bytes:

```
LTA>  tf 0f

16 bit hex value to push on parameter stack:  0000  (1st baseline)

LTA>  tf 0f

16 bit hex value to push on parameter stack:  0005  (last + 1)

LTA>  tf 0f

16 bit hex value to push on parameter stack:  00ff  (mask)

LTA>  tf 0c

0=bank  reading  01 = bsline  re-start seed = D5  **OK**
```

After passing the initial parameters using the tf 0f command and then invoking the test with tf 0c, a status line similar to that shown above will update as the test runs. The bank number will toggle between 0 and 1. The next field will toggle between reading and writing. The next field will count through the specified range of baselines. The seed value relates to the state of the pseudo random generator used as the data source. The **OK** field indicates that there are no errors. If a large number of baselines is being tested, the test takes a long time and the **OK** field will not update until all baselines have been checked. If errors are found, "oops" will appear somewhere in the line to indicate that there have been errors. If errors occur due to problems in a single DRAM path, the mask can be changed to select fewer bytes of the dram for test. For example, a mask of 0001 will test only the least significant byte of the imaginary word. A mask of 0080 will test only the most significant byte of the real word.

4.5 LTA Assembly Language Source Files

The 87C51 source files are listed below. The drawing number 56000D020 was assigned for the source files resulting in the file names below. (D020RO = ROM D020RA = RAM)

Source File	Description
D020RO01.ASM	Initialization
D020RO02.ASM	ROM monitor routines (display memory etc.)
D020RO03.ASM	ROM based HCB handler, read and write memory
D020RO04.ASM	Invoke ROM based tests; (see d020ro05)
D020RO05.ASM	ROM tests 0 and 1; t0 = read/write test of lookup Table rams; t1 = load tables with default contents
D020RA01.ASM	ROM to RAM linkage
D020RA02.ASM	Help screen
D020RA03.ASM	RAM based tests (tf 0c etc.)
D020RA04.ASM	Invoke ram based HCB handler
D020RA05.ASM	RAM based HCB handler (rd/wr DRAM, load Xilinx etc.)
D020RA06.ASM	General observation control

This page intentionally blank.

5 APPENDIX I LIST OF FILES

5.1 Master Control Card (MCC)

5.1.1 Files maintained in corrdwgs/mcc/sch/SCCS

FILE NAMES	DESCRIPTION	NR FILES
L027D01.SCH - L027D04.SCH	MCC schematic	4
L027D01.LIB, L027D01.SRC	Archived Orcad library for MCC	2
Z012D01.LAY - Z012D02.LAY	MCC IC layout	2
Z012D01.LIB, Z012D01.SRC	Archived Orcad library for lay	2
L027.PRT	Bill of materials for MCC	1
W008T01.NET	Netlist for MCC (not updated)	1
N001930C.REV	Updated wirelist for ww company	1
W008T02B	Rev B list of Pin 1 locations	1
W008T03	Ground Tie list	1
MCC.BLK	MCC block diagram	1
V007D01.TIM	System Timing, 131 MSEC CYCLES	1
MCCTOP.PAL	Root sheet for MCC PALs	1
PAL1F.SCH	add: 1F,2F,2G; 3F,4F,4G etc.	1
PAL2C.SCH	val storage adr cntrs 2C,3C	1
PAL3B.SCH	val dly line ctrl 3B,2B	1
PAL4B.SCH	decoder 4B	1
PAL5B.SCH	decoder 5B	1
PAL5C.SCH	data val int bank sel 5C	1
PAL6B.SCH	decoder 6B	1
PAL6C.SCH	decoder 6C	1
PAL7B.SCH	pulsar dly line ctrl 7B,8B	1
PAL8C.SCH	val storage adr cntrs 8C,9C	1
PAL9B.SCH	pulsar dly line ctrl 9B	1
PAL11C.SCH	decoder 11C	1
PAL13B.SCH	baseline counter 13B	1
PAL14G.SCH	sequencer cntr control 14G	1
PAL15B.SCH	8 bit counter 15B,14B	1
PAL16E.SCH	fft cycle ctrl decoder 16E	1
PAL16F.SCH	fringe & int cycle adr 16F	1
PAL16G.SCH	fft cycle seq adr cntr 16G	1
PAL16I.SCH	fifo control 16I	1
PAL19B.SCH	decoder 19B	1
PAL19F.SCH	timing signal decoder 19F	1
PAL20B.SCH	decoder 20B	1
PAL20F.SCH	fringe & int cycle decode 20F	1
PAL21I.SCH	HCB controller	1
PAL22F.SCH	phase lock loop ref div 22F	1
PAL23B.SCH	data valid delay ram ctrl 23B	1
PAL23F.SCH	phase lock loop var radix 23F	1
PAL25F.SCH	test frame formatter 25F	1
PAL28F.SCH	header ram adr counter 28F	1
PAL28H.SCH	CRC control sig generator 28H	1
PAL29J.SCH	misc uP logic 29J	1
PAL30C.SCH	dual 5 to 1 mux 30C,30B	1
PAL30E.SCH	4 to 1 30E,29A,29B,29C,30A	1
PAL32H.SCH	random data generator 32H	1
		52 total

5.1.2 87C51 source files maintained in vlbsoft/mccasm/SCCS

FILE NAMES	DESCRIPTION	NR FILES
Makefile	creates mccrom.hex, mccram.hex	1
*.asm	see SOURCES list in Makefile	15
d025mcc.hcb	HCB protocol	1
mccmem.doc	MCC memory assignments	1
		18 total

5.1.3 PAL files maintained in vlbsoft/pal_prom/pals/mcc/SCCS

FILE NAMES	DESCRIPTION	NR FILES
PAL1F.ABL	used at 1F, 3F, 5F, 7F, 9F, 11F	1
PAL2B.ABL, PAL2C.ABL	2B and 2C	2
PAL2F.ABL	used at 2F,2G,4F,4G,6F,6G, 8F,8G,10F,10G,12F,12G	1
PAL3B,3C,4B,5B,5C,6B,6C,7B,8B	Nine .abl files	9
PAL8C.ABL	used at 8C and 9C	1
PAL9B,11C,13B,14G	Four .abl files	4
PAL15B.ABL	used at 15B and 14B	1
PAL15F,15G,16E,16F,16G,16I	Six .abl files	6
PAL19B,19F,20B,20F,21I,22F	Six .abl files	6
PAL23B,23F,25F,28F,28H,29J	Six .abl files	6
PAL30C.ABL	used at 30C and 30B	1
PAL30E.ABL	used at 30E,29A,29B,29C,30A	1
PAL32H.ABL	32H	1
(40 files cover 63 PALs, there are 34 pal schematics)		40 total

FILE NAMES	DESCRIPTION	NR FILES
*.jed	one jed file for each .ABL	40 total

5.1.4 Sequencer source files maintained in vlbsoft/pal_prom/proms/mccseq/SCCS

FILE NAMES	DESCRIPTION	NR FILES
Makefile	creates rom17g.hex, rom18g.hex rom19g.hex, rom20g.hex, rom12a.hex, rom17b.hex, rom1718f.hex and rom32f.hex	1
*.asm	rom17g.asm, rom18g.asm, rom19g.asm, rom20g.asm, rom12a.asm, rom17b.asm, rom1718f.asm, rom32f0.asm, rom32f1.asm and rom32f2.asm	10
seq0.doc seq1.doc seq2.doc	cryptic files related to the sequencer bit assignments	3
		14 total

5.2 Multiply Accumulate Card (MAC)

5.2.1 Files maintained in corrdwgs/mac/sch/SCCS

FILE NAMES	DESCRIPTION	NR FILES
L003D01.SCH - L003D04.SCH	MAC schematic	4
L003D01.LIB, L003D01.SRC	Archived Orcad library	2
Z003D01.LAY	MAC component layout	1
Z003D02.LAY	MAC comp layout specific to asmbly	1
Z003D01.LIB, Z003D01.SRC	Archived Orcad library for layout	2
K029D01.BLK	Block diag of FFT to MAC intf	1
K030D01.BLK	Block diagram of MAC	1
V009D01.TIM	MAC card edge timing	1
B003T01.PRT	MAC bill of materials	1
L003d01.INC	Orcad include file for BOM	1
W003T01.NET	Netlist for MAC card	1
		16 total

5.3 Long Term Accumulator (LTA)

5.3.1 Files maintained in corrdwgs/lta/sch/SCCS

FILE NAMES	DESCRIPTION	NR FILES
L024D01.SCH - L024D04.SCH	LTA schematic	4
L024D05.DOC - L024D10.DOC	Block diagram and timing etc.	6
L024D11.XIL - L024D15.XIL	Xilinx documentation	5
Z015D01.LAY - Z015D06.LAY	IC layout, adapter layouts	6
L024D01.LIB, L024D01.SRC	Archived library for above	2
		23 total

5.3.2 87C51 source files maintained in vlbsoft/ltaasm/SCCS

FILE NAMES	DESCRIPTION	NR FILES
Makefile	creates ltarom.hex, trcram.hex	1
*.asm	see SOURCES list in Makefile	11
d020asm.hdr	header file for .asm sources	1
d024lta.hcb	HCB protocol	1
d020mem.map	memory assignments	1
		15 total

5.3.3 C source maintained in vlbsoft/ltaasm/SCCS

FILE NAMES	DESCRIPTION	NR FILES
ltabe5.c, ltabetb.c	LTA backend processors	2
?.c	8 other misc backend processors used over the years	8
		10

5.3.4 PAL files maintained in vlbsoft/pal_prom/pals/lta/SCCS

FILE NAMES	DESCRIPTION	NR FILES
D017T2J.ABL - D017T20D.ABL	ABEL source files for LTA loc 2J,4B,5B,6C,7C,8F,9B,10A,10B, 10C,11A,11G,12D,12E,13A,13C, 14C,15C,17B,17E,18B,19C,19D, 20C,20D	25
D017T2J.JED - D017T20D.JED	JED files for the above PALs	25
		50 total

5.3.5 Sequencer files maintained in vlbsoft/pal_prom/proms/ltaseq

FILE NAMES	DESCRIPTION	NR FILES
d019.c	utility to generate sequencer prom images	1
d019ieee.c	utility to generate prom image for 15,15,6 conversion proms at loc 21G and 22G	1
d019ieee.bin	binary image for 21G and 22G	1
d019t7b.src - d019t18d.src	input source files processed by d019.c for loc 7b,8b,13d, 14d,15e,16d,17d,18d	8
d019t7b.bin - d019t18d.bin	binary images for above proms	8
		19 total

5.3.6 Xilinx files maintained in vlbsoft/xilinx/lta

FILE NAMES	DESCRIPTION	NR FILES
mkltamcs.DOC	General comments of interest	1
d018lta1.mac - d018lta4.mac	macro files executed in XACT	4
d018lta1.lca	LCA file generated from above macros	1
d018lt01.lca	Final LCA file after APR	1
d018lta1.cst	Constraint file used above	1
d018lt01.bit	Bit file produced by Makebits	1
d018lt01.mcs	Final HEX file used in correlator	1
d018lt01.rpt	Report file from APR	1
lt01dly.rpt	Report file from XDELAY -worstcase	1
		12 total

This page intentionally blank.

6 Appendix II HCB Protocol for MCC

HCB PROTOCOL FOR THE MASTER CONTROL CARD NRAO DRAWING #A56000D025
TEXT FILE D025MCC.HCB

VLBA CORRELATOR PROJECT
NATIONAL RADIO ASTRONOMY OBSERVATORY
CHARLOTTESVILLE, VA 22903
-

Revision History	Date	By
Preliminary	12/18/91	RPE
Rev A	1/21/92	RPE
Rev B	1/30/92	RPE
Rev C	4/05/92	RPE

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)
TEST FRAME RAM DATA	10 RR TT DD----DD	TEST
TEST FRAME RAM CHECKSUM	11 RR [2 BYTES RETURNED]	TEST
TEST FRAME CONTROL	12 0F FF 0M	TEST
TEST FRAME TIME INCREMENT	13 TT TT	TEST
MASTER/SLAVE FLAG & CLK DLY	14 DD DD	INIT (1)
ASIC CONTROL WORDS	15 AA DD----DD (128 BYTES OF DATA)	INIT (2)
ASIC CHECK SUM	16 [2 BYTES RETURNED]	INIT
ASIC CONTROL WORD STROBE	17 ZZ (ZZ = DUMMY BYTE)	INIT (1)
DATA VALID MUX PROG BITS	18 MM MM MM MM MM TT	INIT (2)
SET FRAME TIME	19 JJ JS SS SS ss ss 00	TEST
CRT PROMPT	1A AA----AA 00----00 (16 BYTES)	INIT (1)
SNAPSHOT REQUEST	20 BB	TEST
SNAPSHOT DATA	21 0X [GETS 1280 BYTES IN RETURN]	TEST
SERVO TEST FRAME	22 0F	TEST
START OBSERVATION	23 ZZ	OBS (1)
END OBSERVATION	24 ZZ (ZZ = DUMMY BYTE)	OBS
ASIC CONTROL WORD TEST	25 AA DD	TEST
SELF TEST	29 PP	TEST

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

The below 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.

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 labeled 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 given where minimal data is transferred.

INCLUDE FILE NAME	FC	TIME (ms)	RATE (KBytes/sec)
FRAMEBUF	0x10	TEST	
FRAMECHK	0x11	TEST	
MCCFRAMECTRL	0x12	TEST	
MCCFRAMEINCR	0x13	TEST	
MCCDELAY	0x14	INIT (1)	
MACCTRLWORD	0x15	16	8
MACCTRLWRDCHK	0x16	Fast since just reads byte in memory	
STROBEMAC	0x17	0.149	
MCCDVALIDMUX	0x18	0.206	29.1
MCCFRAMETIME	0x19	TEST	
MCCPROMPT	0x1a	INIT	
MCCTAKESNAP	0x20	TEST	
MCCSENDSNAP	0x21	TEST	
MCCSERVO	0x22	TEST	
MCCSTARTOBS	0x23	17.7	
MCCENDOBS	0x24	110.0	
MACSHIFTBYTE	0x25	TEST	
MCCSELFTEST	0x29	0.131	

```
**************************************************************************
*
checksum is of the form

                AAAAAAAA AAAAAAAA     accumulated sum
              + 00000000 BBBBBBBB     current byte being summed into checksum
              -------------------
                AAAAAAAA AAAAAAAA     new accumulator sum
**************************************************************************
```

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.

2) TEST FRAME HEADER/DATA RAM CONTENTS

```
RAM:                    10 RR TT DD----DD
RAM CHECKSUM:           11 RR
```

where

RAM	RR	TT	XFER COUNT
HEADER	00	00	20
DATA	01	00 THRU 31	1024 EACH

The check sum is made from information stored in the RAMs. Only the first 12 bytes of the header RAM are used in calculating the checksum. All 32,768 bytes of the data RAM go into it's check sum.

3) TEST FRAME CONTROL

```
TEST FRAME CONTROL:     12 0F FF 0M
```

where

0F FF is the PLL frequency
FFF = 091, FOR NORMAL OPERATION

0M is a mode byte = 0000 0tdf
t = 1, frame time runs
t = 0, frame time is stopped
d = 1, RAM data is in frame
d = 0, random data is in frame
f = 1, frame is VLBA format
f = 0, frame is MKIII format

4) TEST FRAME TIME INCREMENT

```
TIME STEP:              13 TT TT
```

where

TT TT is the frame to frame increase in the header time stamp (in BCD).

MODE	TT TT
VLBA 8-MHZ	00 25
VLBA 4-MHZ	00 50
VLBA 2-MHZ	01 00
MKIII	00 50

5) MASTER/SLAVE FLAG AND DELAY PROGRAM WORDS

DELAY PROG: 14 DD DD

where (for DD DD = m0aa abbb 00cc cddd)

m=0	if the MCC is a slave control card
m=1	if the MCC is a master control card
aaa	clock delay parameter for the MCC FFT INIT signal
bbb	clock delay parameter for the LTA FFT INIT signal
ccc	clock delay parameter for the FCC FFT INIT signal
ddd	clock delay parameter for the PBI FFT INIT signal

6) ASIC CONTROL WORDS

ASIC CONTROL WORDS:	15 AA DD----DD (128 BYTES OF DATA)
ASIC CHECK SUM:	16 [2 BYTES RETURNED]
ASIC CONTROL WORD STROBE:	17 ZZ (ZZ = DUMMY BYTE)

where

AA is a card assignment

and

DD----DD indicates 128 bytes of data (four bytes per ASIC, bytes within ASIC are sent LS byte first, ASIC words are sent in the reverse order that they occur in the ASIC control word serial string).

7) DATA VALID MUX PROGRAM BITS

DATA VALID MUX PROG BITS: 18 MM0 MM1 MM2 MM3 MM4 TT

where

MM is used to program the MCC data valid multiplexers to mirror the multiplexers on the FFT cards

and

TT is a test byte = 00 normally

Setting TT = 0x0F will force data to be invalid at all baselines for the two channels handled by a MCC. The data will be gated to zero at the FFT cards and the data valid count will be zero.

The MM0 through MM4 bytes are defined as follows:

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
MM0=	stn 3-1	stn 3-0	stn 2-1	stn 2-0	stn 1-1	stn 1-0	stn 0-1	stn 0-0
MM1=	stn 7-1	stn 7-0	stn 6-1	stn 6-0	stn 5-1	stn 5-0	stn 4-1	stn 4-0
MM2=	stn11-1	stn11-0	stn10-1	stn10-0	stn 9-1	stn 9-0	stn 8-1	stn 8-0
MM3=	stn15-1	stn15-0	stn14-1	stn14-0	stn13-1	stn13-0	stn12-1	stn12-0
MM4=	stn19-1	stn19-0	stn18-1	stn18-0	stn17-1	stn17-0	stn16-1	stn16-0

where in each of the above "-1" is MSB of mux adr, "-0" is LSB

e.g. stn0-1 and stn0-0 are the 2^1 and 2^0 multiplexor select bits for FFT station 0, and must be set the same as the multiplexor select bits on the station 0 FFT card;

8) SET FRAME TIME

SET FRAME TIME:	19 JJ JS SS SS ss ss 00 (VLBA)
SET FRAME TIME:	19 YD DD HH MM SS ss s0 (MKIII)

9) CRT PROMPT MESSAGE

CRT PROMPT: 1A 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 "MASTER CONTROL CARD".

10) BASELINE SNAPSHOT REQUEST

SNAPSHOT REQUEST: 20 BB

where

BB	is the baseline number to be captured in the MCC FIFO.
BB = 00 to D1	will get baseline data
BB = D2 or D3	will get validity data
BB = D4	will get pulsar validity data

11) SNAPSHOT DATA

SNAPSHOT DATA: 21 0X [GETS 1280 BYTES IN RETURN]

where

0X = 0 for channel 0 (or 2, 4, 6)
0X = 1 for channel 1 (or 3, 5, 7)

Command is issued at least four 131 msec integration cycles after the baseline snapshot request command and will result in one set of 256 36-bit integration results (in 15, 15, 6) from the baseline requested being transferred. The transfer requires 5-bytes per complex point (see below).

1ST BYTE	0RRR RRRR	7 MS bits of REAL
2ND BYTE	RRRR RRRR	8 LS bits of REAL
3RD BYTE	0III IIII	7 MS bits of IMAGINARY
4TH BYTE	IIII IIII	8 LS bits of IMAGINARY
5TH BYTE	00EE EEEE	6 bits of EXP

12) SERVO TEST FRAME

SERVO TEST FRAME: 22 0F

where

0F = 03	TRACK FREQ = 9.020160 MHZ	VLBA, NO SELF TEST (0.20% HIGH)
0F = 02	TRACK FREQ = 8.983040 MHZ	VLBA, NO SELF TEST (0.20% LOW)
0F = 02	TRACK FREQ = 8.983040 MHZ	VLBA, SELF TEST (0.18% HIGH)
0F = 04	TRACK FREQ = 8.945920 MHZ	VLBA, SELF TEST (0.23% LOW)
0F = 04	TRACK FREQ = 8.945920 MHZ	MK3, NO SELF TEST (0.17% HIGH)
0F = 06	TRACK FREQ = 8.908800 MHZ	MK3, NO SELF TEST (0.24% LOW)
0F = 06	TRACK FREQ = 8.908800 MHZ	MK3, SELF TEST (0.15% HIGH)
0F = 07	TRACK FREQ = 8.871680 MHZ	MK3, SELF TEST (0.27% LOW)
0F = 00	TRACK FREQ = 9.001600 MHZ	SPARE VLBA (0.00083% LOW)
0F = 01	TRACK FREQ = 9.038720 MHZ	SPARE VLBA (0.41% HIGH)

where the exact track bit frequencies are

MODE	SELF TEST	FREQUENCY	COMMENT
VLBA	NO	9.0016744 MHZ	(=8MHZ * 22,680/20,000 * 512/516)
VLBA	YES	8.9666484 MHZ	(=8MHZ * 22,680/20,000 * 512/516 * 256/257)
MK3	NO	8.9302325 MHZ	(=8MHZ * 22,500/20,000 * 512/516)
MK3	YES	8.8954845 MHZ	(=8MHZ * 22,500/20,000 * 512/516 * 256/257)

13) OBSERVATION CONTROL

START OBSERVATION:	23 ZZ
END OBSERVATION:	24 ZZ (ZZ = DUMMY BYTE)

14) ASIC CONTROL WORD TEST

ASIC CONTROL WORD TEST: 25 AA DD

where

AA is a card assignment

DD indicates byte of data

15) SELF TEST

SELF TEST: 29 PP

where

PP is the spectral point number to be tested (0-255)

7 Appendix III HCB Protocol for LTA

```
LTA HCB PROTOCOL              NRAO DRAWING #A56000D024
                              TEXT FILE    D024LTA.HCB

VLBA CORRELATOR PROJECT                    Maintained under SCCS in the
NATIONAL RADIO ASTRONOMY OBSERVATORY       ltaasm component
CHARLOTTESVILLE, VA  22903

I)  SUMMARY OF ALL FUNCTION CODES

All function codes below are in HEX:

TRANSFER TYPE                 PROTOCOL                          FREQ OF OCCURRENCE
----------------------------- --------------------------------- ------------------
NOP:                          00 ZZ  (ZZ = RQRD DUMMY BYTE)            TEST
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 rtrnd]     INIT (1)

READDRAM:                     10 SB EB SR SR ER ER SU                  TEST (4)
WRITEDRAM:                    11 SB EB SR SR ER ER SU DD----DD         TEST
XILINX  FIRST TRANSFER:              12 DD----DD (1389 BYTE TRANSFER)  INIT (1)
XILINX  SECOND TRANSFER:      13 DD----DD (1389 BYTE TRANSFER)         INIT (1)
XILINX  GO/NO-GO FLAG:        14          (1 BYTE RETURNED)            INIT (1)
ACCUMTEST:                    15 NN NN BB                              TEST

DWNLD RAM LOOKUP TBL LATER    20 OH OL FF BB PH PL RR AM AL CM CL DD-DD  OBS  (2)
DWNLD RAM LOOKUP TBL NOW      21 OH OL FF BB PH PL RR AM AL CM CL DD-DD  TEST
REQUEST STATUS BYTES          22 (2 bytes returned)                     OBS
REQUEST CHKSUM BUFFER SECTION 23 OH OL NM NL                            OBS
SELECT BANK OF LOOKUP TABLES  25 BB                                     OBS

LTA_OBSMODE                   30 MD                                     INIT (1)
LTA_ACCUMTICS                 31 AC TM TL                               OBS  (2)
LTA_STARTOBS                  32 AC                                     OBS  (2)
LTA_STOPOBS                   33 AC                                     OBS  (2)
LTA_DATA_SOURCE               34 SR                                     TEST
LTA_TIC_CNTRS                 35  (16 bytes rtrnd, tic cntrs 0-7)      TEST
LTA_SELFTEST                  36 ZZ  (ZZ = required dummy byte)         SELFTEST

-------------------------------------------------------------------------------
(1) MODE INDEPENDENT COMMAND
(2) MODE DEPENDENT COMMAND
(3) OPERATIONAL COMMAND
(4) PROVIDES ALTERNATE PATH TO REAL TIME SYSTEM WHEN NO FIR AVAILABLE
*******************************************************************************
Checksum is of the form:

       AAAAAAAA AAAAAAAA      ACCUMULATED SUM
     + 00000000 BBBBBBBB      CURRENT BYTE BEING SUMMED INTO CHECKSUM
     --------------------
       AAAAAAAA AAAAAAAA      NEW ACCUMULATED SUM

*******************************************************************************

List of Appendices:

        Appendix I      RAM LOOKUP TABLE DETAILS
        Appendix II     RAM DOWNLOAD CHECKSUM BUFFER DETAILS
        Appendix III    TRANSFER TIMES
```

II) DETAILED DESCRIPTION OF EACH FUNCTION CODE

All function codes below are in HEX:

1) NOP and MAIN MEMORY OPERATIONS, FUNCTION CODES 00, 01, 02 and 03:

```
   NOP:                  00 ZZ  (ZZ = REQUIRED 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 rtrnd, MS byte first)
```

WHERE
00 00 AA AA is a 32 bit start address and CC CC is a transfer count with MS byte first.

2) READ/WRITE DRAM MEMORY, FUNCTION CODES 10 and 11:

```
   READDRAM:             10 SB EB SR SR ER ER AC
```

WHERE

```
   SB= START BASELINE  (0-255  DECIMAL)
   EB= END   BASELINE  (0-255  DECIMAL)
   SR= START RESULT    (0-2047 DECIMAL, MS BYTE FIRST)
   ER= END RESULT      (0-2047 DECIMAL, MS BYTE FIRST)
   AC= ACCUM CODE      (0-7)  (determines which LTA bank is accessed)
```

THE NUMBER OF BYTES RETURNED IS: (EB+1-SB)*(ER+1-SR)*8 IN THE FOLLOWING ORDER FOR EACH COMPLEX RESULT: (number of bytes must be <= FIFOSIZE)
THE DATA IS READ FROM THE BANK THAT IS NOT BEING "ACCUMULATED" INTO;
(the ACCUM CODE value determines which of the 8 possible bank switch bits is of interest)

Byte #	Description
0	Real byte 3 (MS real byte)
1	Real byte 2
2	Real byte 1
3	Real byte 0 (LS real byte)
4	Imag byte 3 (MS imag byte)
5	Imag byte 2
6	Imag byte 1
7	Imag byte 0 (LS imag byte)

```
   WRITEDRAM:            11 SB EB SR SR ER ER AC DD----DD
```

WHERE

```
   SB= START BASELINE  (0-255  DECIMAL)
   EB= END   BASELINE  (0-255  DECIMAL)
   SR= START RESULT    (0-2047 DECIMAL, MS BYTE FIRST)
   ER= END RESULT      (0-2047 DECIMAL, MS BYTE FIRST)
   AC= ACCUM CODE      (0-7)
```

THE NUMBER OF BYTES (DD----DD) SENT IS: (EB+1-SB)*(ER+1-SR)*8 IN THE FOLLOWING ORDER FOR EACH COMPLEX RESULT: (number of bytes must be <= FIFOSIZE-8)
THE DATA IS WRITTEN TO THE BANK THAT IS NOT BEING "ACCUMULATED" INTO

Byte #	Description
0	Real byte 3 (MS real byte)
1	Real byte 2
2	Real byte 1
3	Real byte 0 (LS real byte)
4	Imag byte 3 (MS imag byte)
5	Imag byte 2
6	Imag byte 1
7	Imag byte 0 (LS imag byte)

3) XILINX XC-3030 PERSONALITY, FUNCTION CODES 12, 13 and 14:

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

4) ACCUMULATION TEST, FUNCTION CODE 15:

```
ACCUMTEST:             15 NN NN BB
```

WHERE NN NN IS THE NUMBER OF 131 MS CYCLES TO ACCUMULATE FOR, MS BYTE FIRST
BB IS THE BANK IN WHICH TO ACCUMULATE (0 or 1)

5) FUNCTIONS ASSOCIATED WITH RAM LOOKUP TABLES:

```
DWNLD RAM LOOKUP TBL LATER:      20 OH OL FF BB PH PL RR AM AL CM CL DD----DD
DWNLD RAM LOOKUP TBL NOW:        21 OH OL FF BB PH PL RR AM AL CM CL DD----DD
```

(See Appendix I for additional details of lookup table contents)
(See Appendix II for ram download checksum buffer details)

Func 20 = hold data in local storage until next 131 ms tic, then download;
(Func 30 must be used to place LTA in observing mode first)

Func 21 = download immediately

```
WHERE RR SPECIFIES THE LOOKUP TABLE RAM
      AM AL SPECIFIES THE START ADDRESS (AM= MS  AL= LS)
      CM CL SPECIFIES THE NR OF BYTES   (CM= MS  CL= LS)

      See Appendix II for OH OL FF BB PH and PL definitions
```

MAXIMUM ALLOWED BYTE COUNT IS PRESENTLY 256
TRANSFERS MUST ALL BE WITHIN THE LS ADDRESS FIELD
THE BYTE COUNT MUST BE EXACTLY 256 IF THE CHECKSUM IS GOING TO BE USED

The following RR codes provide full access to every byte of every table:

TABLE OF RR VALUES:

RR	RAM
00	BERSLTLS (back end result, LS)
01	BERSLTMS (back end result, MS)
02	FERSLTLS (front end result, LS)
03	FERSLTMS (front end result, MS)
04	FEBSLN (front end bsline)
05	FE_AC_MAP (front end accum_time & map)
06	FEV_AC (front end validity accum_time)
07	BE_AC_MAP (back end accum_time & map)
08	BEV_AC (back end validity accum_time)

The accum_time lookups relate to the bank switching. We need to accumulate into one bank while reading from the opposite bank. Thus at the front end, we use the FE_AC_MAP lookup to determine which of the 8 possible bank switch bits is of interest for a specific baseline. (At the back end, we use the BE_AC_MAP lookup.) The map lookups are similiar, in that the front end and back end result mapping tables have 8 possible maps to use for normal baselines.

These protocols will be used as follows:

While observing:

In a given 131 ms interval, up to two Function Code 20 transfers may be performed. The blocks of data will be stored in 8751 local memory space. Immediately after the next 131 ms tic, the data will be moved from local storage to the target ram chip.

When not observing:

Any practical number of Function Code 21 transfers may be performed. The blocks of data will still be stored initially in 8751 local memory

space, but at the end of the HCB transaction, the data will immediately be transferred to the target ram chip.

```
REQUEST STATUS BYTES:        22 (2 bytes returned)
REQUEST CHKSUM BUF SECTION:  23 OH OL NM NL (NM*256 + NL bytes returned)

SELECT RAM BANK              25 BB
```

Func 22 = first byte returned = nr of times a HCB transfer has crossed the 131 msec tic boundary (normal tic jobs are skipped if this happens so this must always be zero)

second byte returned = nr of times the 131 msec tic jobs over ran the allotted time extending into the LTA accumulation time, this must always be zero

Func 23 = return the checksum table entries from the checksum table starting at offset OH OL in the table; NM NL bytes returned; it is expected that all entries associated with a single RR code will be requested in each request; the offset into the table for each RR code is given in the following table, along with the NM NL byte count:

RR	OFFSET	NN	Bytes returned (NM= ms byte of count) (NL= ls byte of count)
0	0	128	128*8= 1024
1	128*8	128	128*8= 1024
2	256*8	128	128*8= 1024
3	384*8	128	128*8= 1024
4	512*8	4	4*8= 32
5	516*8	4	4*8= 32
6	520*8	8	8*8= 64
7	528*8	4	4*8= 32
8	532*8	8	8*8= 64

(See Appendix II for details of the checksum buffer)

Func 25 = select the active bank (0-3) for the following ram lookup tables:

FEBSLN
FE_AC_MAP
FEV_AC
BE_AC_MAP
BEV_AC

where BB= 0-3 to select the bank. This bank becomes effective during the next 131 ms interval.

6) OBSERVATION CONTROL, FUNCTION CODES 30, 31, 32, 33, 34, 35:

```
LTA_OBSMODE      30 MD
LTA_ACCUMTICS    31 AC TM TL
LTA_STARTOBS     32 AC
LTA_STOPOBS      33 AC
LTA_DATA_SOURCE  34 SR
LTA_TIC_CNTRS    35     (16 bytes returned, tic counters 0-7)
                        (cntr 0 first, LS byte first)
LTA_SELFTEST     36 ZZ  (ZZ = required dummy byte)          SELFTEST
```

where MD = 1 to set observing mode on, 0 to set it off
AC = accumulation code 0-7
TM = ms byte of number of 131 ms tics for accumulation time
TL = ls byte of number of 131 ms tics for accumulation time
SR = 0 to select normal data from MCC
= 1 to select built in pseudo random data generator

Func 30 places the LTA in or out of observation mode
When observing mode is off, the 131 ms tic interrupt handler stays in rom. When on, the handler branches into ram. Likewise, the main idle loop stays in rom when off and branches into ram when on. Note that Func 32 turns observing mode on also.

Func 31 places the nr of tics in storage for use by Func 32

Func 32 begins observing for code AC at the next tic, using the TM TL
that was last received under Func 31;
(Starts the tic counter off at zero, clear accum and use bank 0)
Also places the LTA in observation mode in case Func 30 has not
been invoked prior to Func 32.

Func 33 stops observing for code AC at the next tic
(complements the bank bit so last results are available and sets
the maximum accumulation time)

Func 34 selects the LTA input data path

Func 35 requests a readback of the 8 tic counters

Func 36 specifies a self test cycle is to be run

APPENDIX I

RAM LOOKUP TABLE DETAILS

NOTE: As described in Appendix II, in order to use the checksum mechanism for checking the table downloads, each transfer block must be exactly 256 bytes long, on 256 byte boundaries. Thus for cases where the transfer counts are shown as 0-209 or 0-215 etc., the actual transfer should pad the count out to 256 bytes total.

1) BERSLT (RR= 00 and 01)

The Back End Result table does a translation from one 11 bit result number (0-2047) to another.

There are 16 possible maps. Maps 0-7 are available for normal data. Maps 8, 10, 12 and 14 are available for tape validities. Maps 9, 11, 13 and 15 are available for Pulsar validities. These assignments result from the fact that the MS bit of the map number is used in the hardware to switch the accumulation code selection from the normal lookup tables to the special validity lookup tables. The LS bit of the map number is then used to determine if the tape or pulsar sections of the validity lookup tables is selected.

This table was planned originally to provide translation for any restraints imposed by the Fast Page Mode of the DRAMs. This function has now been moved to the output of the FIR. See comments below about expected future uses for this table.

The table is two bytes wide. The transfer of the table contents over the HCB is done on a byte basis, with 256 byte blocks per transfer. Each 256 byte block transfer starts at zero in the LS field of the AMAL address (below) and continues through 255.

BERSLTLS (RR= 00, LS byte of the two byte table)
Bytes 0-32767 contain the lower 8 bits of the translated result number. The 15 bit address to the ram is composed of three fields, represented by the AMAL address in the HCB header as follows:

AMAL ADDRESS			DATA BYTES
AM7		AL0	As the AMAL address result
MS FIELD	MIDDLE	LS FIELD	field counts from the starting
	MS RSLT	LS RSLT	point, the 256 data bytes transferred are the LS bytes
MAP NR	BITS	BITS	of the result translation to
[14..11]	[10..8]	[7..0]	write in the lookup table.
0-15	0-7	0-255	

(11 BIT RESULT FIELD COUNTS 0-2047)

The 11 LS bits of the AMAL address above corresponds to the result number requested from the FIR. The data bytes transferred over the HCB provide a translation pointer for reading the results from the LTA.

The MAP NR field counts through the 16 possible mapping tables, (8 for normal data and 8 for validity data).

BERSLTMS (RR= 01, MS byte of the two byte wide table)
Bytes 0-32767 contain the upper 3 bits of the translated result number. The 3 LS bits of the byte contain the 3 bits of interest. The upper 5 bits are "don't care". The address fields to the ram are as defined above for RR = 00.

Presently, only a 1-to-1 mapping is expected to be used for all back end maps.

It is likely that once we understand the operational requirements for handling validities, this table may become more useful. In particular, if arrays are added together the validities for those arrays will also be added together in the LTA but the FIR may need multiple copies of the result of adding together the validities. Thus this output re-ordering table may be used to provide the multiple copies from a single validity.

2) FERSLT (RR= 02 and 03)

The Front End Result table does a lookup from MAC result number (8 arrays of 256 results each = 2048 results) to LTA result number. There are 16 possible maps. Maps 0-7 are available for normal data. Maps 8, 10, 12 and 14 are available for tape validities. Maps 9, 11, 13 and 15 are available for Pulsar validities.

This table determines, for every MAC result, which LTA result the MAC result will be accumulated into. Thus results may be re-ordered and they may be reduced by accumulating more than one MAC result into the same LTA result. The table is two bytes wide.

The MS byte of the table contents serves a dual function. The three LS bits are part of the LTA result number. The next four bits are presently spare. The MS bit is used to control the Clear Accumulator (CLAC) function. This bit is gated in a PAL with the Master CLAC signals for the 8 possible accumulation times. This bit provides the CLAC Inhibit function required when data reduction is occurring. When this bit is zero, it allows the Master CLAC to be applied as the current MAC result is being accumulated into the LTA. When this bit is set to logic one, it inhibits the application of the MASTER CLAC. See d034t01.doc for additional discussion. When no data reduction is occurring, this bit is low for all addresses. When data reduction is occurring, this bit is low for the first MAC result that will be accumulated into a specific LTA result location, and is high for each subsequent MAC result that will be accumulated into the same LTA result location.

The transfer of the table contents over the HCB is done on a byte basis, with 256 byte blocks per transfer.
Each 256 byte block transfer starts at zero in the LS field of the AMAL address (below) and continues through 255.

FERSLTLS (RR= 02, LS byte of the two byte wide table)
Bytes 0-32767 contain the lower 8 bits of the LTA result number. The 15 bit address to the ram is composed of three fields, represented by the AMAL address in the HCB header as follows:

AMAL ADDRESS			DATA BYTES
AM7		AL0	
MS FIELD	MIDDLE	LS FIELD	
	MAC	MAC	As the AMAL address result field counts from the starting point, the 256 data bytes transferred are the LS bytes of the result translation to write in the lookup table.
MAP NR	ARRAY	RESULT	
[14..11]	[10..8]	[7..0]	
0-15	0-7	0-255	

The LS FIELD counts through the 256 spectral results from a single MAC array. The MIDDLE FIELD counts through the 8 MAC arrays. The MAP NR FIELD counts through the 16 possible mapping tables, (8 for normal data and 8 for validity data).

FERSLTMS (RR= 03, MS byte of the two byte wide table)
Bytes 0-32767 contain the upper 3 bits of the LTA result number and the CLAC Inhibit bit.
The 3 LS bits of the byte contain the 3 bits of interest.
The next 4 bits are "don't cares". The MS bit is the CLAC Inhibit bit as defined above. The address fields to the ram are as defined above.

The map number field provides 8 tables (0-7) for general use, and 8 tables relating to validities (8-15). Table 0-7 contents vary as a function of observing modes. Tables 8-15 are for validities. If arrays are added together, the associated validities will also need to be added together.

Present plans call for no re-ordering to be done other than for data reduction where more than one MAC result is accumulated into the same LTA result. The mapping from the AMAL address fields above to the table contents is shown below for several different examples:

AMAL address fields											Translates to										
A2	A1	A0	R7	R6	R5	R4	R3	R2	R1	R0	r10	r9	r8	r7	r6	r5	r4	r3	r2	r1	r0
no reduction											R7	R6	R5	R4	R3	R2	R1	R0	A2	A1	A0
2:1 array reduction												R7	R6	R5	R4	R3	R2	R1	R0	A1	A0
4:1 array reduction													R7	R6	R5	R4	R3	R2	R1	R0	A0
8:1 array reduction														R7	R6	R5	R4	R3	R2	R1	R0
2:1 transform reduction												R6	R5	R4	R3	R2	R1	R0	A2	A1	A0
4:1 transform reduction													R5	R4	R3	R2	R1	R0	A2	A1	A0
8:1 transform reduction														R4	R3	R2	R1	R0	A2	A1	A0
2:1 array, 2:1 transform reduction													R6	R5	R4	R3	R2	R1	R0	A1	A0
8:1 array, 8:1 transform reduction																	R4	R3	R2	R1	R0

3) FEBSLN (RR= 04)

The Front End Baseline table does a lookup from the Hardware Baseline (HBL) to the LTA baseline. The table is one byte wide. There are four banks available.
The AMAL address in the HCB header includes the bank address bits as follows:

AM7----------AL0		DATA BYTES
MS FIELD	LS FIELD	As the AMAL address field counts from
BANK NR	HARDWARE	the starting point, the first 216 data
	BASELINE	bytes transferred are the Hardware
[9..8]	[7..0]	Baseline to LTA baseline translations.
0-3	0-215	(pad the 0-215 out to 0-255)

The LS FIELD counts through the hardware baselines, padded out to 255. The BANK NR FIELD counts through the four "global" banks that are available. (HCB function 0x25 selects one of four banks globally for the FEBSLN, FE_AC_MAP, FEV_AC, BE_AC_MAP and BEV_AC tables, meaning that the bank selection is common to the five tables.)

Baseline translations for HBLs 0-209 must remain in the range 0-209. Baseline translations for HBLs 210-215 must be one to one.

Baselines 0-209 are normal data, 210-212 are validity data and 213-215 are non-existent data but require one to one mapping since the hardware operates as if they were actual data sources, and thus baselines 213-215 in the LTA memory are not available for any other use. Baselines 216 - 255 are don't cares.

4) FE_AC_MAP (RR= 05)

The Front End AC_MAP table does a lookup from the Hardware Baseline (HBL) to the FRONT END ACCUMULATION TIME and FRONT END MAPPING codes.

The AMAL address in the HCB header includes the bank address bits as follows:

AM7----------AL0		DATA BYTES
MS FIELD	LS FIELD	As the AMAL address field counts from
BANK NR	HARDWARE	the starting point, the first 216 data
	BASELINE	bytes transferred contain the ACCUMULATION
[9..8]	[7..0]	and MAPPING codes for each baseline.
0-3	0-215	(pad the 0-215 out to 0-255)

The LS FIELD counts through the hardware baselines, padded out to 255. The BANK NR FIELD counts through the four "global" banks that are available. (HCB function 0x25 selects one of four banks globally for the FEBSLN, FE_AC_MAP, FEV_AC, BE_AC_MAP and BEV_AC tables, meaning that the bank selection is common to the five tables.)

The bit definitions are:

7	6	5	4	3	2	1	0
A3	A2	A1	A0	M3	M2	M1	M0

where A[3..0] is the accumulation time code
selects one of 8 accumulation times in the LTA for normal results (codes 0-7), and is a don't care for tape and pulsar validity results

and M[3..0] is the mapping table code
selects one of 8 result mapping tables at LTA input

for normal results (codes 0-7), or one of 8 mapping
tables for validities (codes 8-15);
M3 = 1 forces accumulation time to be determined from
the FEV_AC lookup for tape and pulsar validity results
(codes 8-15)

ACCUMULATION and MAPPING codes for HBLs 0-209 must be in the range 0-7.

For HBLs 210-215, the ACCUMULATION code is a don't care. The MAPPING code must be as follows:

HBL	MAP codes	Description
210	8,10,12 or 14	Tape Data Validities
211	9,11,13 or 15	Pulsar Validitys (first half)
212	9,11,13 or 15	Pulsar Validitys (second half)
213	don't care	
214	don't care	
215	don't care	

The MSB of the MAPPING code being = 1 is used to force the accumulation lookup code to come from the FEV_AC lookup table.
The MAP codes associated with tape validities must be even.
The MAP codes associated with pulsar validities must be odd.

5) FEV_AC (RR= 06)

The Front End Validity AC lookup table provides the FRONT END ACCUMULATION TIME code for each tape or pulsar validity in order to determine which bank to accumulate into.

Bytes 0-209 are the ACCUMULATION TIME codes for each baseline. The table is one byte wide. There are four banks available. The AMAL address in the HCB header includes the bank address bits as follows:

```
AM7----------------------AL0
----------------------------
MS FIELD  MIDDLE    LS FIELD
--------  --------  --------
BANK NR   VALIDITY  HBL or     (tape validities) or
          TYPE      SPECRSLT   (pulsar validities)
[10..9]     [8]     [7..0]
-------   --------  --------
  0-3       0-1     0-209 or  (hardware baselines, pad to 0-255)
                    0-255     (spectral result)
```

Where the validity type is:

0= Tape Validities
1= Pulsar Validities

For tape validities, the LS field is counting through baselines 0-209 as indicated above, and the bit definitions of the contents are:

```
 7  6  5  4  3  2  1  0
-- -- -- -- -- -- -- --
 0  0  0  0  0  A2 A1 A0
```

where A[2..0] is the ACCUMULATION TIME CODE, 0-7. Each of the baselines 0-209 may be assigned to any of the 8 available codes, and should be identical to those in the FE_AC_MAP table for the same baseline.

For pulsar validities, A[2..0] should be set to the same value for all 256 spectral results, where the value should be one of 0-7, and should be equal to the ACCUMULATION TIME CODE in the FE_AC_MAP table for the baselines involved. All baselines involved with the pulsar observation must have the same ACCUMULATION TIME CODE.

6) BE_AC_MAP (RR= 07)

The Back End AC_MAP table does a lookup from the Hardware Baseline (HBL) to the BACK END ACCUMULATION TIME and BACK END MAPPING codes.

There are four banks available.
The AMAL address in the HCB header includes the bank address bits as follows:

AM7----------AL0		DATA BYTES
MS FIELD	LS FIELD	As the AMAL address field counts from
BANK NR	HARDWARE BASELINE	the starting point, the first 216 data bytes transferred contain the ACCUMULATION and MAPPING codes for each baseline.
[9..8]	[7..0]	
0-3	0-215	(pad the 0-215 out to 0-255)

The LS FIELD counts through the hardware baselines, padded out to 255. The BANK NR FIELD counts through the four "global" banks that are available. (HCB function 0x25 selects one of four banks globally for the FEBSLN, FE_AC_MAP, FEV_AC, BE_AC_MAP and BEV_AC tables, meaning that the bank selection is common to the five tables.)

Bytes 0-215 contain the ACCUMULATION and MAPPING codes for each baseline.

The bit definitions are:

7	6	5	4	3	2	1	0
A3	A2	A1	A0	M3	M2	M1	M0

where A[3..0] is the accumulation time code
selects one of 8 accumulation times in the LTA for normal results (codes 0-7), and is a don't care for tape and pulsar validity results

and M[3..0] is the mapping table code
selects one of 8 result mapping tables at LTA input for normal results (codes 0-7), or one of 8 result mapping tables for validities (codes 8-15);
M3 = 1 forces accumulation time to be determined from the BEV_AC lookup for tape and pulsar validity results (codes 8-15)

ACCUMULATION and MAPPING codes for HBLs 0-209 must be in the range 0-7.

For HBLs 210-215, the ACCUMULATION code is a don't care. The MAPPING code must be as follows:

HBL	MAP codes	Description
210	8,10,12 or 14	Tape Data Validities
211	9,11,13 or 15	Pulsar Validitys (first half)
212	9,11,13 or 15	Pulsar Validitys (second half)
213	don't care	
214	don't care	
215	don't care	

The MSB of the MAPPING code being = 1 is used to force the accumulation lookup code to come from the BEV_AC lookup table.
The MAP codes associated with tape validities must be even.
The MAP codes associated with pulsar validities must be odd.

7) BEV_AC (RR= 08)

The Back End Validity AC lookup table provides the BACKEND ACCUMULATION TIME CODE for validities being read out to the FIR in order to determine which DRAM bank to read the validities from.

The AMAL address in the HCB header includes the bank address bits as follows:

```
AM7----------------------AL0
----------------------------
MS FIELD  MIDDLE    LS FIELD
--------  --------  --------
BANK NR   VALIDITY  HBL or     (tape validities) or
          TYPE      SPECRSLT   (pulsar validities)
[10..9]     [8]     [7..0]
-------   --------  --------
  0-3       0-1      0-209 or  (hardware baselines, pad to 0-255)
                     0-255     (spectral result)
```

Where the validity type is:

0= Tape Validities
1= Pulsar Validities

For tape validities, the LS field is counting through baselines 0-209 as indicated above, and the bit definitions of the contents are:

7	6	5	4	3	2	1	0
0	0	0	0	0	A2	A1	A0

where A[2..0] is the ACCUMULATION TIME CODE, 0-7. Each of the baselines 0-209 may be assigned to any of the 8 available codes, and should be identical to those in the FE_AC_MAP table for the same baseline.

For pulsar validities, A[2..0] should be set to the same value for all 256 spectral results, where the value should be one of 0-7, and should be equal to the ACCUMULATION TIME CODE in the FE_AC_MAP table for the baselines involved. Thus all baselines involved with the pulsar observation must have the same ACCUMULATION TIME CODE.

APPENDIX II

RAM DOWNLOAD CHECKSUM BUFFER DETAILS

1) Each transfer of up to 256 bytes to the LTA will be accompanied with a checksum word. The LTA will provide a table of entries for each 256 byte block of each downloaded table. Each entry will contain eight bytes as follows:

BYTE	FUNCTION
0	Flag byte
1	Block number
2	CHKSUM PRED HI BYTE
3	CHKSUM PRED LO BYTE
4	CHKSUM ACTUAL HI BYTE
5	CHKSUM ACTIAL LO BYTE
6	SPARE
7	SPARE

The flag byte states represent the following conditions:
(The MS nibble of the FLAG byte is equal to the RR code.)

FLAG BYTE LS NIBBLE	MEANING
0	This entry not in use
1	PRED has been entered, need to generate ACTUAL
2	ACTUAL generated, checked OK and stored
4	ACTUAL generated, checked BAD and stored

The checksum buffer will contain 540 of these eight byte entries, or 4320 bytes total. The order of these entries is a function of the RR codes as follows:

RAM NAME	RR	ADDRESS ORGANIZATION	#256 BYTE BLOCKS	BASE OFFSET IN BYTES INTO CHKSUM TABLE
BERSLTLS	0	16 x 8 x 256	128	0
BERSLTMS	1	16 x 8 x 256	128	128 x 8
FERSLTLS	2	16 x 8 x 256	128	256 x 8
FERSLTMS	3	16 x 8 x 256	128	384 x 8
FEBSLN	4	4 x 256	4	512 x 8
FE_AC_MAP	5	4 x 256	4	516 x 8
FEV_AC	6	4 x 2 x 256	8	520 x 8
BE_AC_MAP	7	4 x 256	4	528 x 8
BEV_AC	8	4 x 2 x 256	8	532 x 8

Thus the first 128 eight byte entries in the checksum buffer represent the 128 blocks of 256 bytes each in the BERSLTLS ram etc.

Refer to Appendix I for more details of the address field organization of each of the above rams. The entries in the checksum buffer are in the same order as the natural counting order of the address fields as described in Appendix I.

The checksum buffer will initially be cleared (flag byte =0). When a table download is received over the HCB, the protocol will include the PRED checksum for the transferred buffer, and the offset into the checksum buffer for the associated 5 byte entry. (This has not been reflected as of 10-15-91 in the definitions of Function Codes 20 and 21 above.) The new protocol will be defined here first, then copied above when stabilized:

DOWNLOAD RAM LOOKUP TABLES:

FUNCTION CODE 20 OH OL FF BB PH PL RR AM AL CM CL DD----DD
FUNCTION CODE 21 OH OL FF BB PH PL RR AM AL CM CL DD----DD

WHERE:

```
OH OL SPECIFIES THE OFFSET INTO THE CHECKSUM TABLE
        (OH= MS  OL= LS)

FF MS NIBBLE = RR (0-8), LS NIBBLE = 1

BB IS THE BLOCKNR
     (0-127, 0-3 OR 0-7 DEPENDENT ON RR)

PH PL IS THE CHECKSUM (PH= MS  PL= LS)

RR SPECIFIES THE LOOKUP TABLE RAM (0-8)

AM AL SPECIFIES THE START ADDRESS (AM= MS  AL= LS)

CM CL SPECIFIES THE NR OF BYTES   (CM= MS  CL= LS)
```

The LTA will store the FF, TT and the PRED checksum at the indicated offset in the checksum table (OH,OL).

During any 131 ms tic interrupt where no table download is pending, the LTA will check the checksum buffer, starting at the beginning and look for the first flag byte where the LS nibble equals 1. For this entry, the LTA will read the associated 256 byte block of the ram and calculate the checksum, store the checksum in the table and set the flag byte according to the result of the comparison with the predicted checksum. The checksum is thus only valid if the original download was exactly 256 bytes and aligned on an appropriate 256 byte boundary. The real time system can inhibit the calculation and check of the checksum by setting the FF byte so the LS nibble is zero.

APPENDIX III

TRANSFER TIMES

General comments:

The total time to transfer a single byte over the HCB is approximately 128 usec. When the size of a transfer reaches 1024 bytes, the time per byte drops to approximately 10 usec/byte. These transfer rates are for simple cases where the target reads the bytes and writes them into local memory, with no additional processing.

HCB transactions are limited to 2048 bytes total in either direction, including the header bytes. Larger transfers are broken up into multiple transfers.

The LTA will not respond to any HCB interrupt during the first approximately 8 msec of each 131 msec interval.

In the LTA, there are no protocol functions where there is a significant delay between the end of the transfer and the point where the target is ready to receive another transfer.

The following measurements mostly use timexN(), with a few oscilloscope measurements included. The measurements are broken down into general categories.

1) NOP:

timexN (hcbNop,ltabus,lta) = 128 usec

This gives a measure of the overhead involved in sending a single byte command.

2) WRITE MEMORY:

timexN (hcbFillMem,0x1000,1024,5,ltabus,lta) = 9.8 msec for 1024 bytes.

This is approximately 10 usec/byte or 100 KByte/sec. This gives a measure of the raw transfer rate when many bytes are transferred over the HCB and just written into memory. The download of ltaram.hex thus will transfer bytes at this rate.

3) CALCULATE CHECKSUM:

timexN (hcbRdChkSum,0x1000,1024,pred,ltabus,lta) = 8.6 msec for 1024 bytes.

This measures the time taken by the LTA to do a checksum and return the result. When pred = the actual value so there is no printf invoked, the 8.6 msec time is measured. The total time of the transaction is thus on the order of 10 usec/byte, although the bytes are not actually transferred over the HCB. But the HCB and the target are occupied for this interval of time.

4) COMBINATION OF WRITE MEMORY AND CALCULATE CHECKSUM:

WRITE MEMORY and CALCULATE CHECKSUM are of interest when downloading ltaram.hex. There are approximately 10K bytes in the binary image.

```
10Kbytes * 10 MSec/Kbyte write to memory = 100 msec for write
10Kbytes * 10 MSec/Kbyte checksum        = 100 msec for checksum
                                           ---
Thus the minimum time  possible is:        200 msec total
```

Measurement of the total time for the hcbSrecLoad() function is:

timexN hcbSrecLoad,"ltaram.hex",0,0,ltabus,lta = 550 msec

These functions are only of interest during initialization.

5) XILINX TRANSFER

timexN hcbLoadXil,"d0181t01.mcs",ltabus,lta,0x12,2778,2,0 = 255 msec

Using a scope to measure the elapsed time between the first interrupt and the final data transfer showed 150 msec total. This is 54 usec per byte, thus this transfer is approximately 5 times slower than the raw transfer rate of the bus.

This function is only of interest during initialization.

6) LOOKUP TABLE DOWNLOADS

period 5,ldferslt shows on the scope that the two blocks of 256 bytes transferred each 131 msec tic take approximately 2 msec each. Thus these transfers occur at the normal HCB raw transfer rate of approximately 10 usec per byte. In order to allow the download of these tables while in observing mode, the restriction of two blocks of 256 bytes per 131 msec tic applies. Thus some table downloads are spread over many tics.

7) OBSERVATION CONTROL FUNCTIONS (function codes 0x30 - 0x33)

These are all small transfers of either two or four bytes total. Thus the transfer time is on the order of 150 usec maximum. (128 usec for the first two bytes plus 10 usec per byte for each additional byte)

8) STATUS CHECKING FUNCTIONS (function codes 0x22 and 0x23)

Status and checksum results are presently readback at low priority every five seconds. The checksum results are the bulk of the transfer, approximately 4300 bytes. These transfers are at the raw bus rate (approximately 10 usec per byte).

9) LTA_SELFTEST (function code 36)
135 usec