UNISYS

1993 5204

V500 EXECUTE MODULE (XM)

Entry/ Medium Systems Group
PASADENA DEVELOPMENT CENTER

UNISYS RESTRICTED

ENGINEERING DESIGN SPECIFICATION

REVISIONS

REV LTR	REVISION ISSUE DATE	PAGES REVISED, ADDED, DELETED OR CHANGE OF CLASSIFICATION	PREPARED BY	APPROVED BY
A	4/17/86 ECN 50340	Initial Issue	A. Gomez G. Kabenjian L. Simpson	R. Young H. Li D. Welbourn J. Chen D. Lau
B	ECN 51995	Implemented changes per REE 099436 as follows: Updated Shift Chain information in Section 7.1. Added Appendix G - Freeze Control Diagnostic Module. Added Section 3.10.6 - XM Write Algorithm. Added MCACM Error Vector Definition in Appendix B.	L. Simpson G. Kabenjian	A. Gomez H. Li J. Chen N. Fleming

PAS 3908 REV. 6-87

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF ILLUSTRATIONS

1 INTRODUCTION

1.1 PREFACE

This system design specification describes the Execute Module (XM) for the V500 Medium Systems processor.

The XM is a general purpose programmable processing module implemented in the V500 as an instruction execution unit.

1.2 RELATED DOCUMENTS

Information contained in the following documents is pertinent to this specification.

SPEC. NO.	NAME
1993 5162	V500 System
1997 5390	V Series Instruction Set
1993 5212	V500 Fetch Module
1993 5220	V500 Memory Control and Cache Module
1993 5238	V500 Memory Module
1993 5253	V500 I/O Memory Concentrator
1993 5303	V500 Maintenance Subsystem
1993 5279	V500 Architecture
1993 5337	Fault Detection Design Recommendations

2 GENERAL DESCRIPTION

The XM or execution module is a module designed to provide the processing functions needed to implement the Medium systems instruction set.

The module is composed of two major parts: the control section and the data section. The partitioning into 2 boards is done, for the most part, as a control board and a data board.

The control section contains the microsequencer with control store and bus control interfaces. Additional logic exists for assisting the microcode in handling exception case testing and instruction startup relating to the Fetch Queue 2 (The control related extension of the Fetch Interface). To balance the two boards, the Fetch Queue 1 is also located on the control card.

The data section contains the data processing functions and the data storage devices needed as scratchpad and I/O buffers. It provides test and status information to the control section and contains the memory requestor.

FIGURE 2-1 XM OVERALL BLOCK DIAGRAM

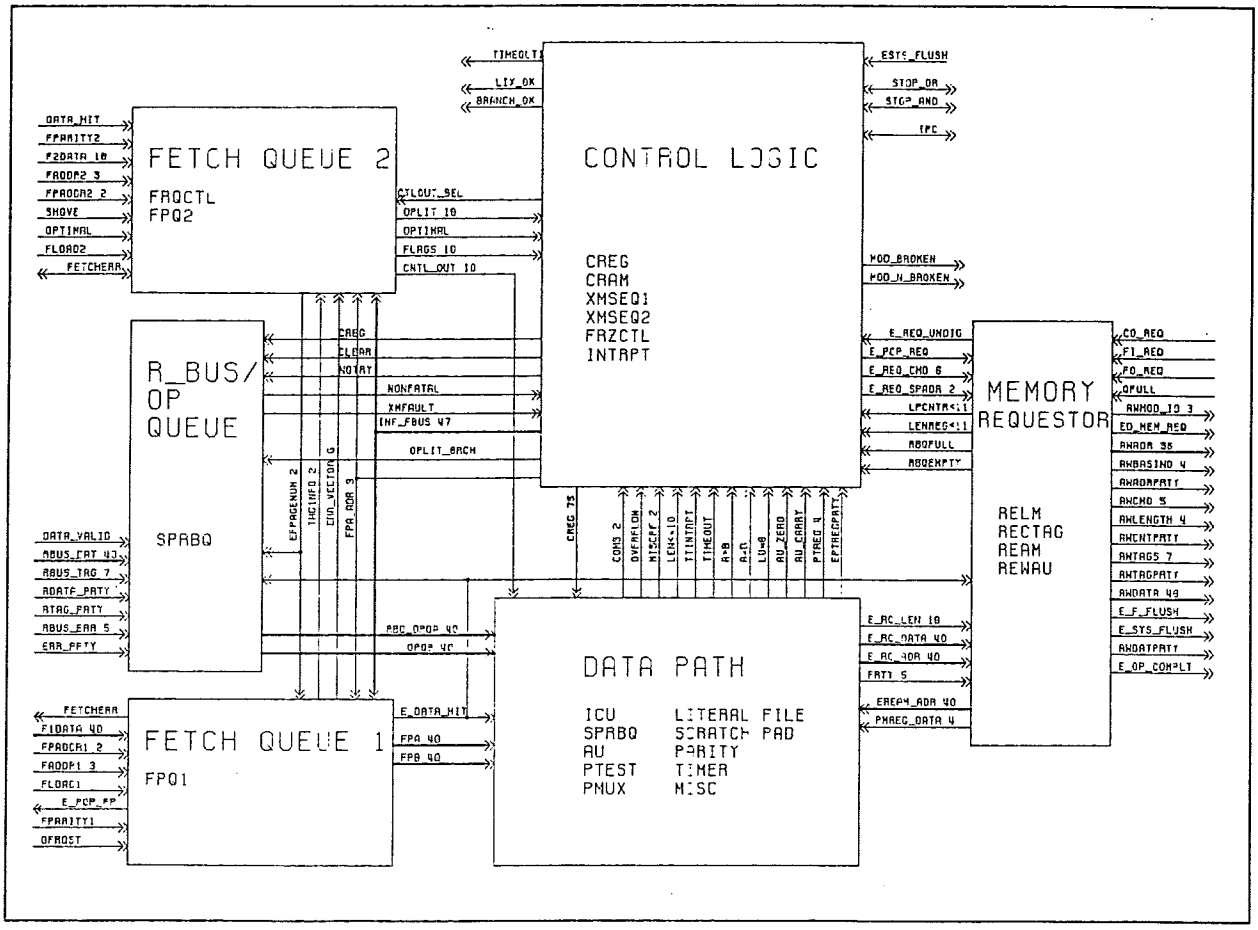
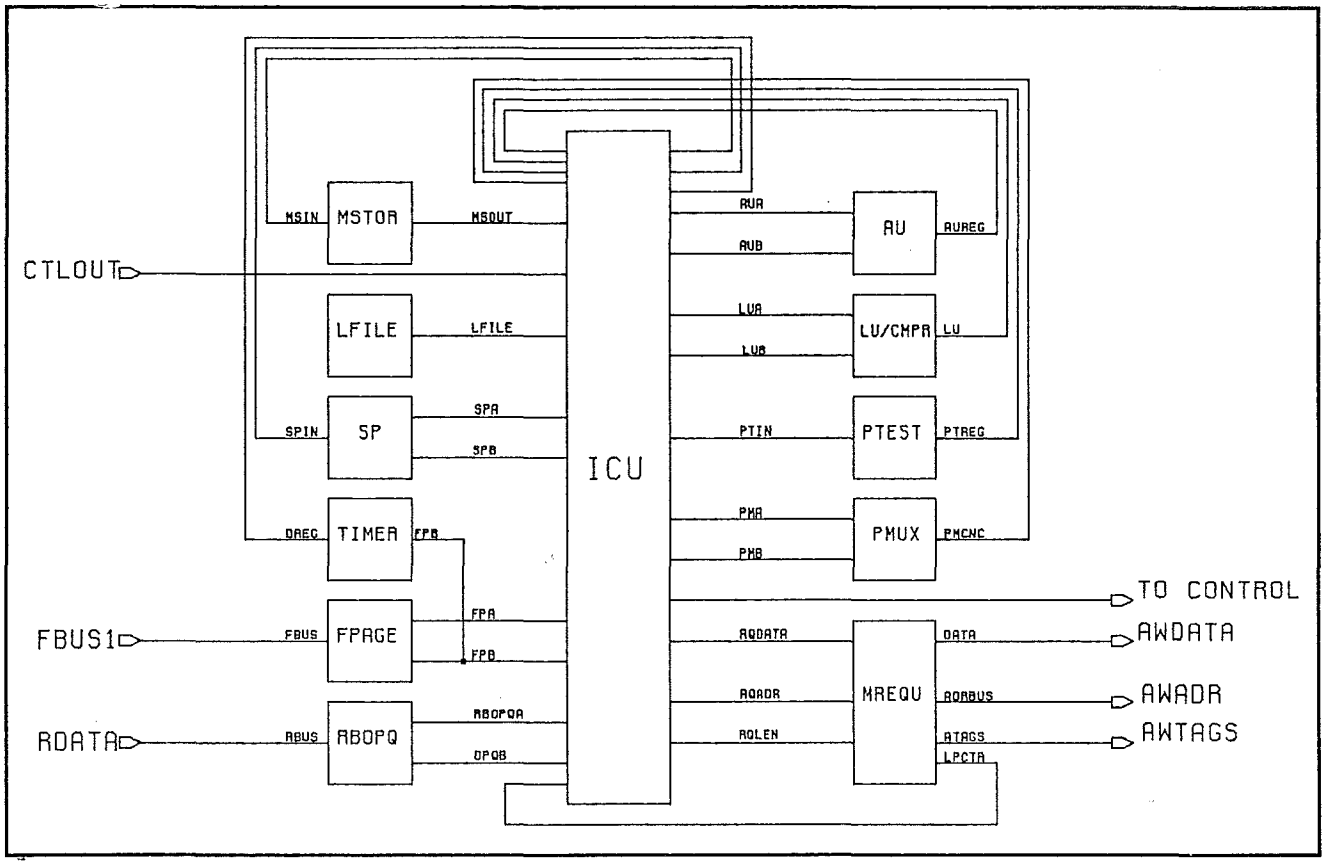


FIGURE 2-2 XM DATA SECTION



3 DATA SECTION

The data section has ten operand sources which feed the InterConnect Unit (ICU) (See Figure 2-2). The operand sources are as follows:

Scratchpad RAM — 16 words, 40 bits wide, dual read, single write from the ICU

Litfile — 256 words, 40 bits wide Read only ram structure for storage of constants: written during control store load.

Fetch Queue 1 — 4 pages of 8 words, 40 bits wide, read only, full read on one port (FPA), limited read on the other (FPB). XM sees only one page at a time. Port FPB is multiplexed with the timer output.

Operand Queue — 4 pages of 2 words each used for receiving operand prefetch information from the Rbus. Its pages correspond to the Fetch pages, only one page is readable at a time.

REUS Queue — an 8 word ordering input queue, 40 bits wide, read only

AU Register — 1 word 40 bits wide:the pipelined output of the Arithmetic Unit

Mass Store — a 256 word, 40 bit wide Ram Structure with one output. Capable of either reading or writing during a clock cycle but not both. The Mass Store entry which is to be accessed is addressed through the MSREG; a register which is loaded from the data section.

DREG — A 40 bit wide general purpose register connected to the ICU.

Timer — An eight digit BCD down counter loaded by the DREG and read ir the FPB.

3 DATA SECTION (Continued)

Requestor — 3 40-bit general purpose storage elements that can be addressed and fed through a common path to the ICU.

Control Entries — 10 bits of data may be sent from the Control Card. These will align with the least significant digits of the ICU, the most significant digit is driven by the PTREG (only when PMASEL=6) and the other bits will be zeroes.

The data operators are as follows:

Logic unit — provides logical operations and test for zero on result.

Comparison Unit — provides full comparison function with flags as well as MSD compare.

Pmux — provides rearrangement of A input data and B input data on a digit basis, and concatenates them to form a 10 digit result.

Ptest — provides combinatorial processing of data producing a 4 bit wide result which feeds the control section or can provide a second level combinatorial function. The PTREG is connected to the MSD of CNTLOUT where it can be manipulated by PMUXA.

Memory Requestor — contains an address handler consisting of three 10 digit decimal counters for use in loops or as address registers, it is loaded from the ICU and drives the ICU and/or the address field of the AWBUS. It also contains a write alignment unit which drives the data field of the AWBUS and is loaded from the ICU. Only one counter is accessable at a time. In addition, there is the length manipulator unit with 3 10-bit registers. These are loaded through the RGLEN mux of the ICU.

3 DATA SECTION (Continued)

Arithmetic unit — provides decimal and hex, addition and subtraction outputting to the AUREG, also provides the control section with an AU_CARRY flag and an AU_ZERO flag.

The data section receives data through the Fetch queues, the operand queue, and the RBUS queue via the memory requestor. It transmits data externally from the operand sources and internally passes data to the control section through the PTREG.

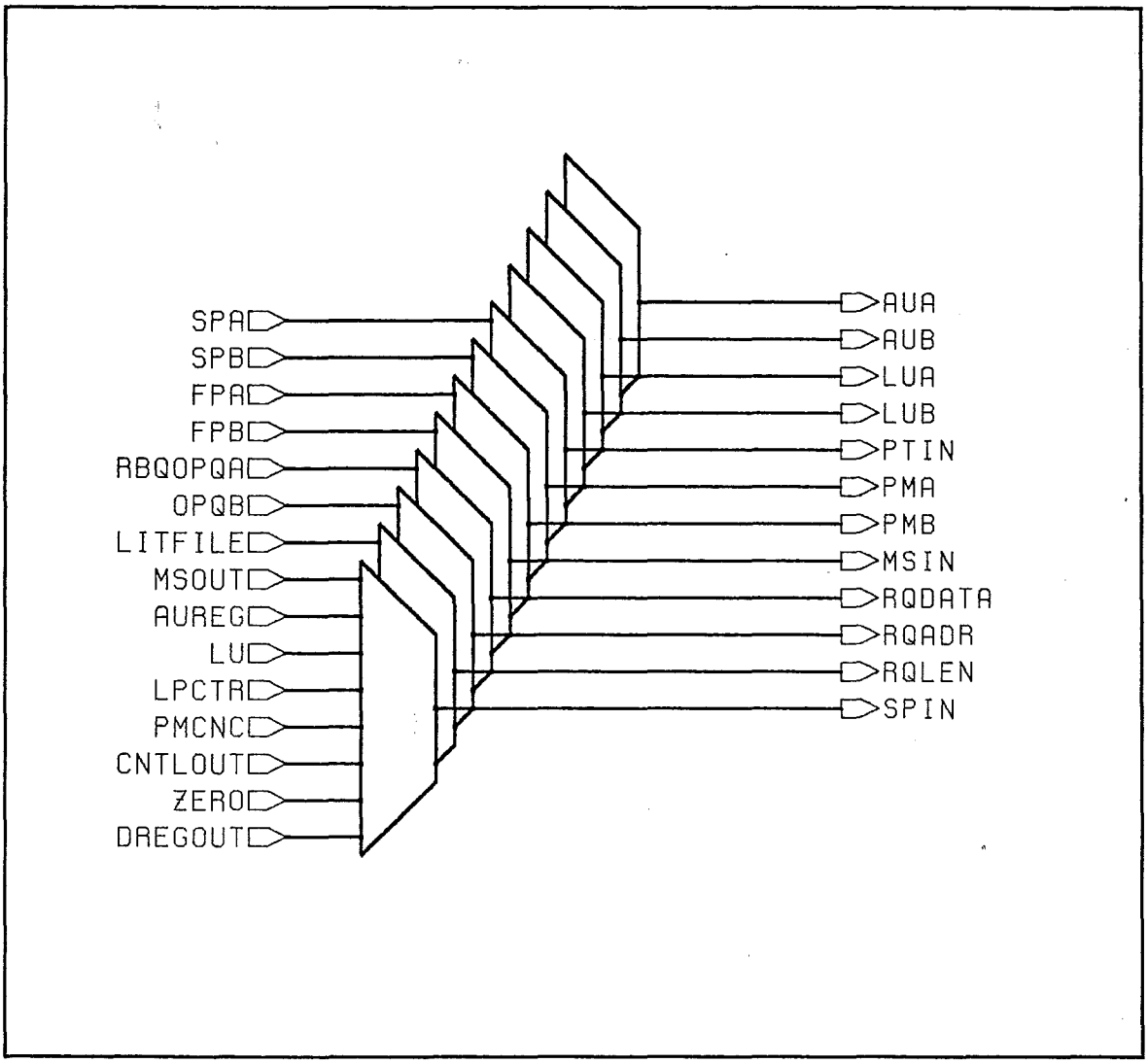
3.1 INTERCONNECT UNIT (ICU)

The Interconnect Unit (ICU) is a firmware controlled crossbar switch that routes selected sources to their destinations. It is controlled by a unique group of select lines for each of the multiplexers associated with either an operator or storage element input. Figure 3-1 is a block diagram of the ICU digit slice.

At the heart of the data path, the ICU receives operands from the RBQ/QFQ, SF, FF, MSTORE, Litfile, DREG, and Memory Requestor. In addition, the results of the LU, AU and FMCNC also feed the ICU. If a source has a dual port read access, then both of these outputs drive the ICU. The cycle time restricts the number of passes through the ICU and limits the path to one processing element between storage devices.

The design of the ICU can be thought of as a set of 1 of n muxes, where n varies for each of the destinations on the ICU. For example, the A-input to the AU (AUA) is fed by SFA, FFA, RBQQFQA, LFILE, MSOUT, AUREG, ZERO, and DREG. Thus, three bits in the CREG select one of these eight sources to drive the AUA data lines. In a similar fashion, the AUB receives one of eight possible inputs selected by the three bit field in the CREG, AUB_SEL. This structure is repeated throughout the rest of the ICU based on table 3-1. At the left, is a list of the possible destinations and at the top, is a list of the sources. Each row includes an indication of the connections currently defined for that particular destination. The select field, to the right of the table, serves as a marker for each of the connections defined and also to present the encoding of each select field.

FIGURE 3-1 INTERCONNECT UNIT (ICU)



DESTINATIONS	Sources: SPA	SFB	FPA	FPB	RBQ/OPQA	OFQE	LIFL	MSUT	AUR E	LUTG	LPTR	FMCA	CNTR	ZERO	DREG OUT	CREG BITS
AUASEL	0	-	1	-	2	-	3	4	5	-	-	-	-	6	7	38:36
AUBSEL	-	0	-	1	-	2	3	4	5	-	-	-	6	-	7	35:33
LUASEL	0	-	1	-	2	-	3	-	-	-	-	-	-	-	-	98:97
LUESEL	-	0	-	1	-	2	3	-	6	-	-	-	4	7	5	96:94
FTSEL	0	-	1	-	2	3	-	4	5	-	-	-	7	-	-	110:108
PMASEL	0	-	1	-	2	-	3	7	4	-	-	-	6	-	5	61:59
PMBSEL	-	0	-	-	1	2	3	-	-	-	-	-	-	-	-	58:57
MSINSEL	0	-	-	-	1	-	-	-	2	-	-	-	-	-	3	82:81
RQDSEL	0	-	1	2	3	-	4	5	6	7	-	-	-	-	-	70:68
RQASEL	0	-	1	2	3	-	4	5	6	7	-	-	-	-	-	67:65
RQLSEL	-	0	-	-	-	-	1	2	3	-	-	-	4	5	-	64:62
SPINSEL	0	-	1	-	2	-	-	3	4	5	6	7	-	-	-	117:115

TABLE 3.1 ICU PATH DEFINITIONS

NOTE : DREGIN receives the RQADR as its input.
PMREGA has the same sources as RQLEN (3:0).
PMREGB is driven by PTREG.
PTSEL Code = 6 is not allowed.

3.2 SCRATCH PAD RAM

The Scratchpad RAM is a dual-port-read, single-port-write 16 word deep RAM with 40 bit wide word. It uses three separate addresses for the one write and two read ports:

A address - designates read to A-output.
B address - designates read to B-output.
C address - designates location to be written from the ICU.

Data input is sourced by SPIN mux in the ICU matrix.

3.3 FETCH QUEUE 1

Fetch Queue 1 contains four pages of 8 words each,40 bits wide. The XM sees it as a read only device which appears only 8 deep. The Fetch Module sees it as a write only port also only 8 deep per page of the queue (See Figure 3-2). Addresses are provided as follows:

READ ADDRESSES:

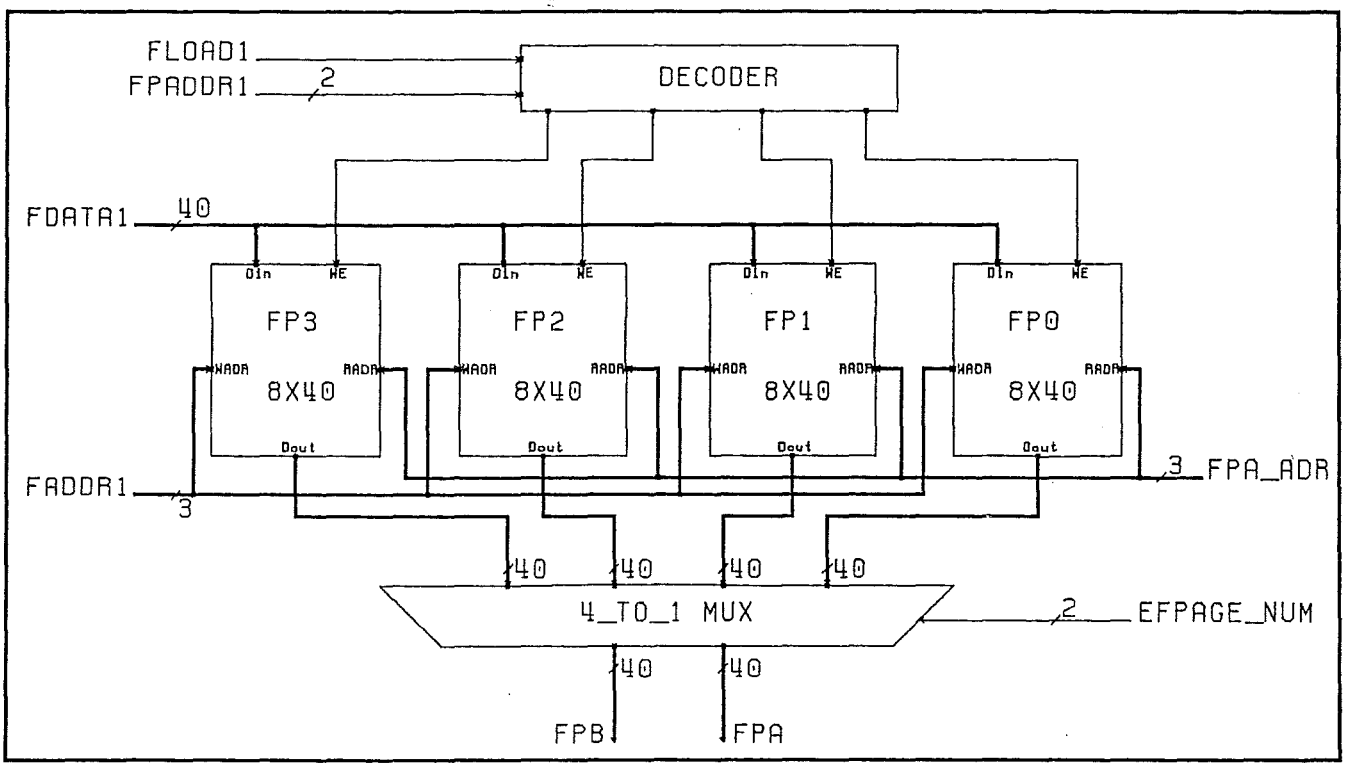
FFPAGE_NUM - specifies which page is to be read
FFA_ADR - selects the entry in the specified page and data is read out on the FFA port.

WRITE ADDRESSES:

FFADDR1 - specifies the page that is to be written.
FADDR1 - selects the entry to be written in the specified page.

Port FFA has access to any entry in the queue, but FFB is restricted to Address 2 only. FFB is shared with the TIMER output. An internal command selects which one of the two is active.

FIGURE 3-2 XM FETCH QUEUE 1



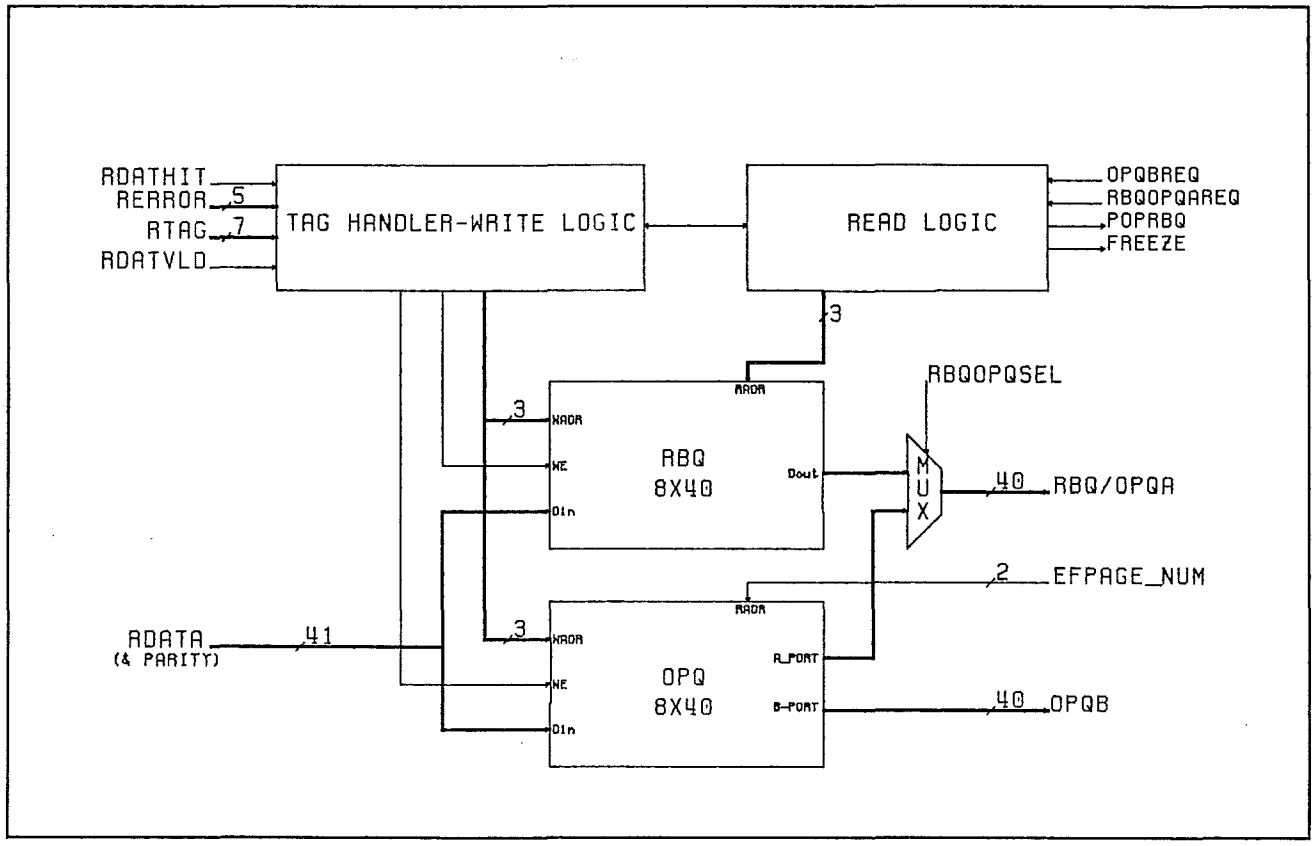
3.4 OPERAND QUEUE

The Operand Queue is a 4 page, 2 entry/page input Queue used to receive Operand information from the Rbus, requested by the Fetch unit. The page address (FFPAGE_NUM) corresponds to the Fetch queue pages and as with the Fetch queue, only one page is visible to the XM. Each page contains two 40 bit entries, an "A" entry and a "B" entry, which are available to the XM on ports REQOPQA and OPQB respectively. A "data valid" bit exists for each entry, which is set upon receipt of the data, and reset on POP_FF, "Data hit", Flush, or an unconditional read. Data remains valid throughout the instruction. OPLIT branch will invalidate the current page entries.

3.5 R-BUS QUEUE

The RBUS queue is an 8 deep sequencing input queue. It is 40 bits wide (See Figure 3-3). What is meant, by a sequencing queue, is that incoming data has a tag which indicates the order in which the data is to be read without respect to the order of its arrival. Entries are invalidated as they are read out. Its output is shared with the "A" entry in the OPQ.

FIGURE 3-3 XM RBUS/OPERAND QUEUE



3.6 PROGRAMMABLE MULTIPLEXER (PMUX)

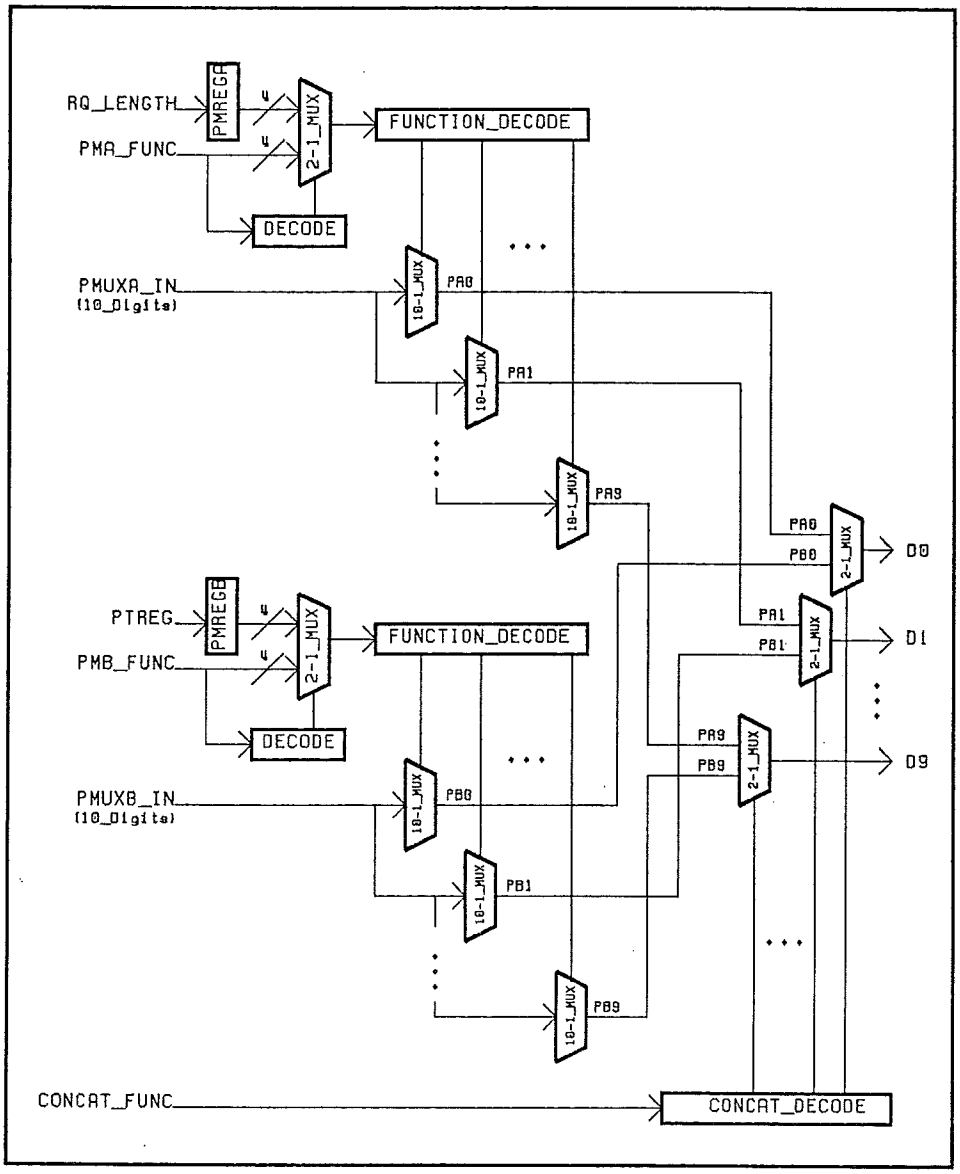
The Pmux performs positional transformation of its input on a digit by digit basis. It receives two 40 bit inputs and outputs a 40 bit word to the ICU (See Figure 3-4).

The Pmux is user programmable with 16 different functions defined. There is a data dependent function register (PMREG), for PMUXA, that is loaded whenever the CREG bit 'PMREGLD' is set. A value of "F" selects the PMREG as the source of the function to be performed. The input to the PMREG is shared with the least significant digit of the memory requestor's length field (RQLEN). In addition, the PMUX is used to concatenate the two 40 bit inputs according to the user's predetermined functions. The result of the concatenation, based on the independantly processed A and B inputs, is referred to as the PMCNC output.

PMUXE has a data deperdent register that is loaded by setting PMREGLD high. The input to this second PMREG is sourced by the PTREG.

The functions performed by either PMUX, whenever their function select is equal to "F", is determined by the contents of their respective PMREG. The value in the PMREG corresponds directly to the functions defined in Section 4.6.6. However, if the contents of the PMREG has a value of "F", an error is set causing a dead freeze condition.

FIGURE 3-4 XM PMUX



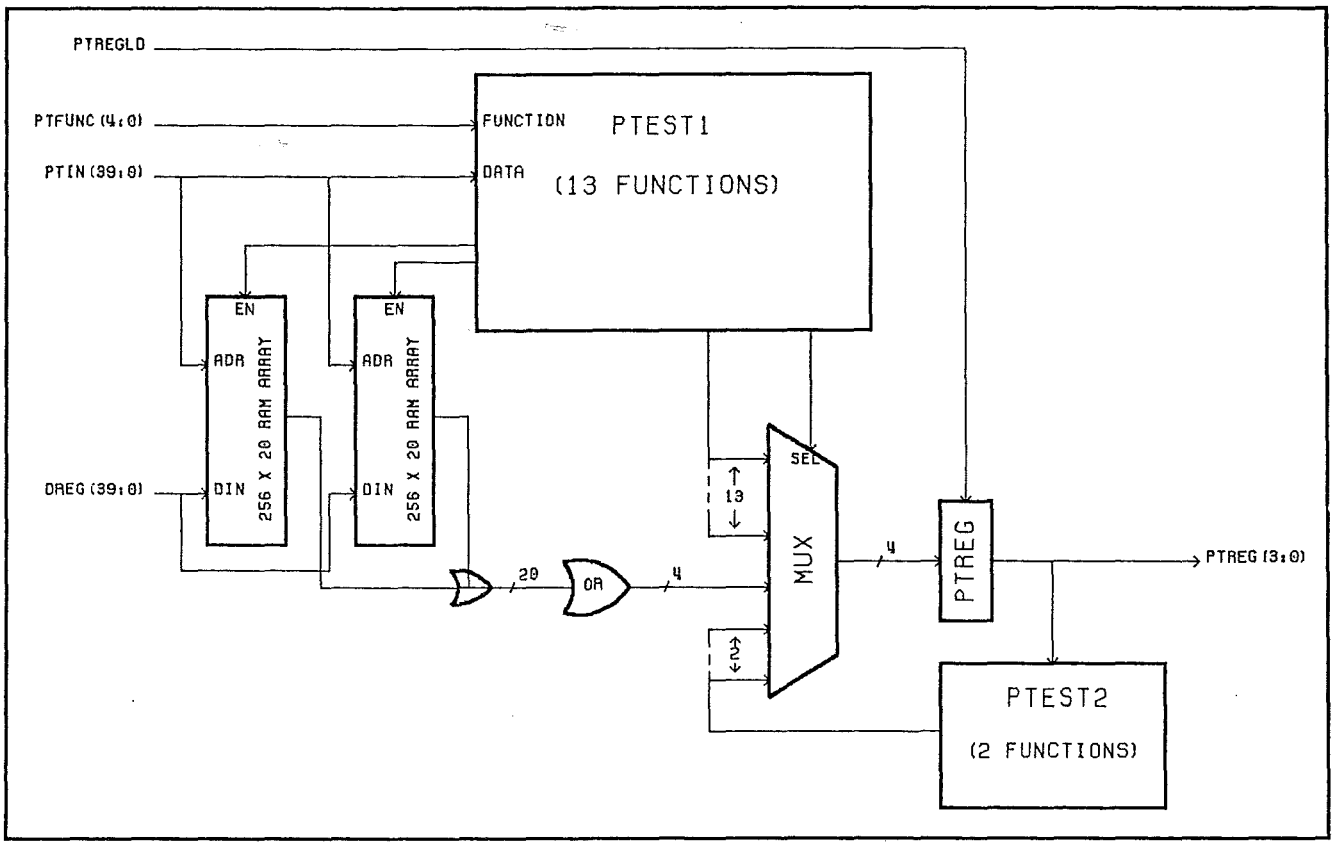
3.7 PROGRAMMABLE TEST LOGIC (PTEST)

The Ptest provides 13 full combinatorial transformations on its input from the ICU. The user may in addition program 2 functions across the 40 bit input with full combinatorial processing of each of the 5 bytes of input. This part of Ptest is called Ptest1, it feeds PTREG(4 bits wide) which sources the control section and Ptest2.

Ptest2 provides combinatorial processing of the PTREG thus allowing greater flexibility across byte boundaries at the cost of extra processing time. Ptest2 has 2 functions and can have its output stored in PTREG or sent to the most significant digit of an ICU input with zeros filling digits 4-8 of the input and CNTLOUT feeding the least significant digits (See Figure 3-5).

For the definition of the Ptest functions, see Appendix A.

FIGURE 3-5 XM PTEST LOGIC



3.8 THE ARITHMETIC UNIT (AU)

The 40 bit 2 input AU is capable of performing both binary and decimal addition and subtraction. For speed purposes, a carry-look-ahead is used across the AU. The AU of the XM requires one clock to perform an arithmetic operation. It has a registered output, thus results can not be used until the next clock. Back to back operations are permitted. (See Figure 3-6.)

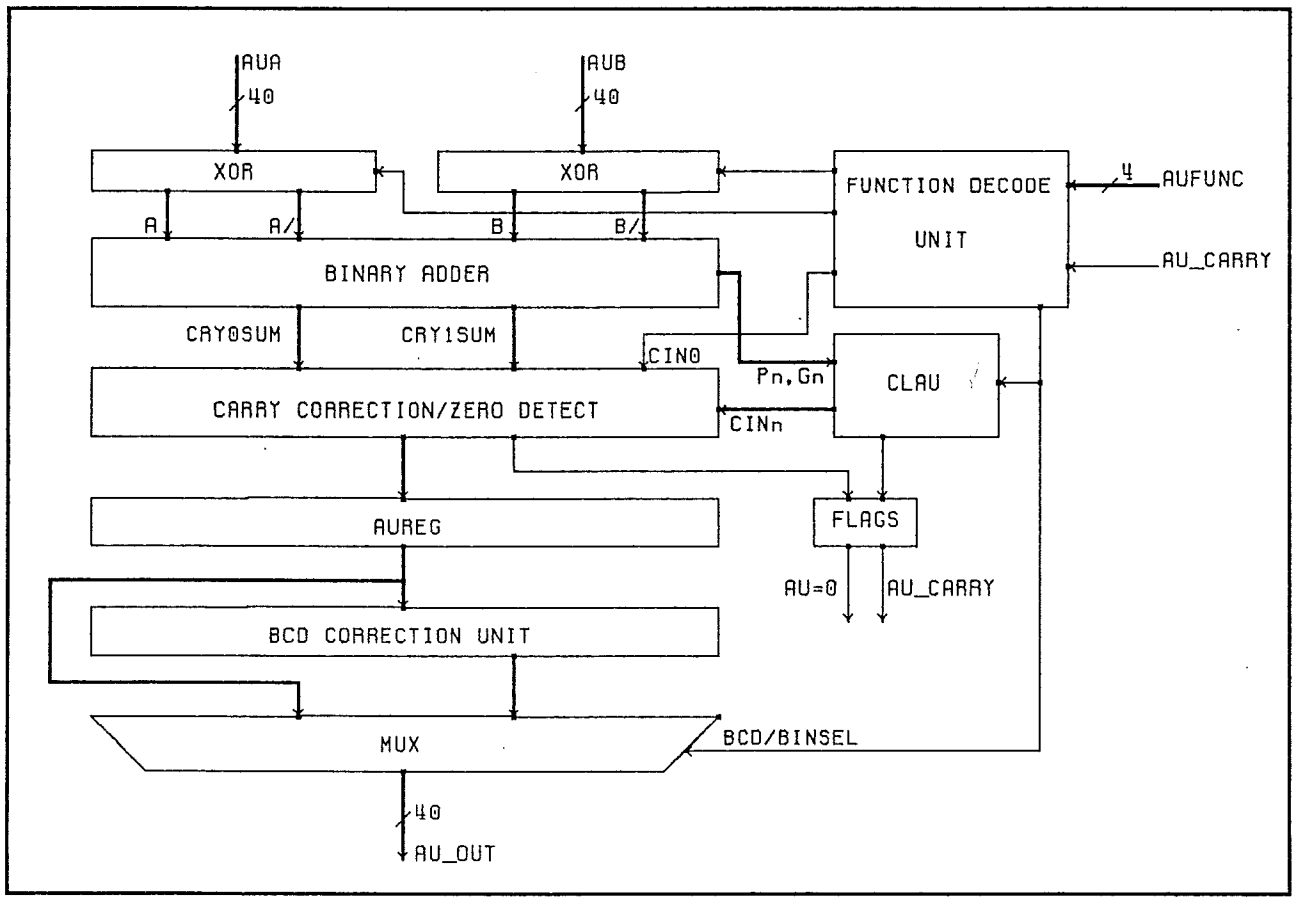
For AU functions, see Section 4.6.7.

The AU provides status to the control section in the form of two status lines. These lines are: AU_ZERO and AU_CARRY (carry out from the most significant digit).

The result of performing decimal arithmetic on non-BCD inputs (i.e. undigits) is contained in Tables 3.2 and 3.3.

Undigit detection at the inputs to the AU, sets a flag which the microcode can branch on.

FIGURE 3-6 XM ARITHMETIC UNIT



3.8 THE ARITHMETIC UNIT (AU) (Continued)

TABLE 3.2 XM AU DIGIT OUTPUT FOR DECIMAL ADDITION (A+B)

	B input (no carry)										A+B (decimal)						
A input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	-
0	0	1	2	3	4	5	6	7	8	9	*0	*1	*2	*3	*4	*5	*6
1	1	2	3	4	5	6	7	8	9	*0	*1	*2	*3	*4	*5	*6	*7
2	2	3	4	5	6	7	8	9	*0	*1	*2	*3	*4	*5	*6	*7	*8
3	3	4	5	6	7	8	9	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9
4	4	5	6	7	8	9	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	*2
5	5	6	7	8	9	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	*2	*3
6	6	7	8	9	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	*2	*3	*C
7	7	8	9	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	*2	*3	*C	*D
8	8	9	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	*2	*3	*C	*D	*6
9	9	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	*2	*3	*C	*D	*6	*7
A	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	*2	*3	*C	*D	*6	*7	*0
B	*1	*2	*3	*4	*5	*6	*7	*8	*9	*2	*3	*C	*D	*6	*7	*0	*1
C	*2	*3	*4	*5	*6	*7	*8	*9	*2	*3	*C	*D	*6	*7	*0	*1	*2
D	*3	*4	*5	*6	*7	*8	*9	*2	*3	*C	*D	*6	*7	*0	*1	*2	*3
E	*4	*5	*6	*7	*8	*9	*2	*3	*C	*D	*6	*7	*0	*1	*2	*3	*4
F	*5	*6	*7	*8	*9	*2	*3	*C	*D	*6	*7	*0	*1	*2	*3	*4	*5
A input	-	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	B input (with carry)																

Note : * indicates carry to next higher digit.

3.8 THE ARITHMETIC UNIT (AU) (Continued)

TABLE 3.3 XM AU DIGIT OUTPUT FOR DECIMAL SUBTRACTION

	MINUEND (with borrow)										B-A (DECIMAL)					
SUBTRAHEND	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	-
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	-
1	*9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	0
2	*8	*9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	1
3	*7	*8	*9	0	1	2	3	4	5	6	7	8	9	A	B	C	2
4	*6	*7	*8	*9	0	1	2	3	4	5	6	7	8	9	A	B	3
5	*5	*6	*7	*8	*9	0	1	2	3	4	5	6	7	8	9	A	4
6	*4	*5	*6	*7	*8	*9	0	1	2	3	4	5	6	7	8	9	5
7	*3	*4	*5	*6	*7	*8	*9	0	1	2	3	4	5	6	7	8	6
8	*2	*3	*4	*5	*6	*7	*8	*9	0	1	2	3	4	5	6	7	7
9	*1	*2	*3	*4	*5	*6	*7	*8	*9	0	1	2	3	4	5	6	8
A	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	0	1	2	3	4	5	9
B	*7	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	0	1	2	3	4	A
C	*6	*7	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	0	1	2	3	B
D	*5	*6	*7	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	0	1	2	C
E	*4	*5	*6	*7	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	0	1	D
F	*3	*4	*5	*6	*7	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	0	E
-	*2	*3	*4	*5	*6	*7	*0	*1	*2	*3	*4	*5	*6	*7	*8	*9	F

NOTE: * indicates borrow from next higher digit

Undigit results will be included for completeness only, they are implementation dependant and may change.

3.9 LOGIC UNIT (LU) and COMPARATOR

The Logic Unit (LU) peforms a Boolean operation on two 40 bit inputs, A and B, and outputs a 40 bit result to the ICU. The operations executed by the LU are: A AND B, A OR B, A XOR B, 'NOT A' and B.

In parallel with the LU, there is a comparator that sets two flags, A > B and A = B, appropriately. There is a flag, LU=0, that gets set if the output of the LU is equal to zero. These three flags are part of the test conditions feeding the sequencer in the control section.

3.10 MEMORY REQUESTOR (Mem Req)

The Memory Requestor is the XMs interface to MCACM via the AWBUS. It increments/decrements address and length as appropriate for Read and Write operations. It handles mod 10 data alignment during Writes and does tag resolution for Reads and Writes. The Memory Requestor is composed of 4 major components (See Figure 3-7): the Address Manipulator, the Length Manipulator, the Write Alignment Unit, and the Control and Tag Generator Unit.

The following 4 sections give brief descriptions of the individual units.

3.10.1 MEM REQ ADDRESS MANIPULATOR (REAM)

The main purpose of the Address Manipulator is to update and hold addresses for r/w memory requests. It consists of 3 ten-digit registers and an inc/dec unit which can operate on all three registers. All 3 registers can be used as general purpose storage, where they can be read out on the LPCTR lines in the ICU Matrix, as well as loop counters, meaning they can increment or decrement by any number from 1 to 10. REAM provides a LPCTR=0 or crossed 0 flag to the Test Condition MUX.

The address manipulator unit also contains an output register with bus drivers, and a parity generator on all 40 bits for driving the memory address field of the AWBUS (See Figure 3-8).

FIGURE 3-7 MEMORY REQUESTOR

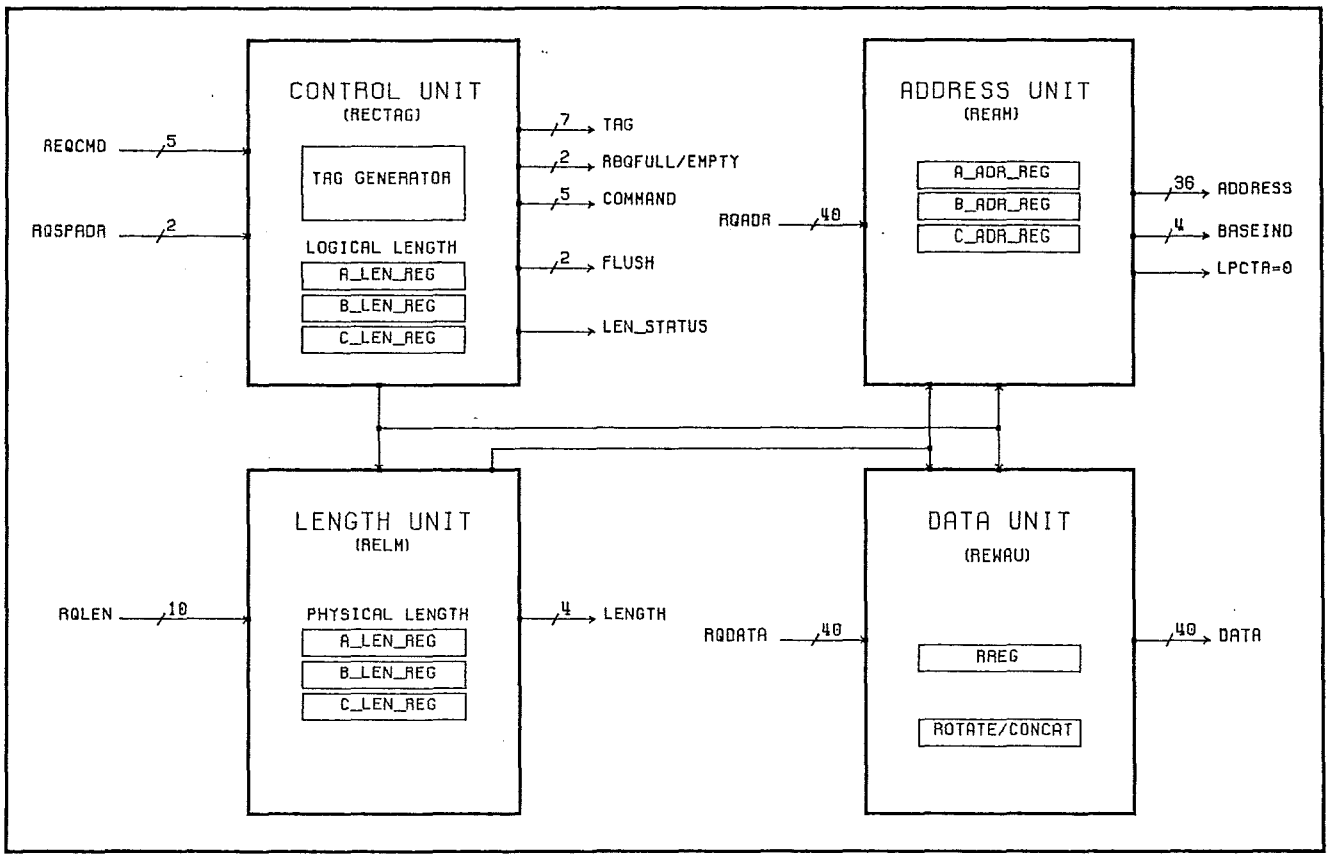
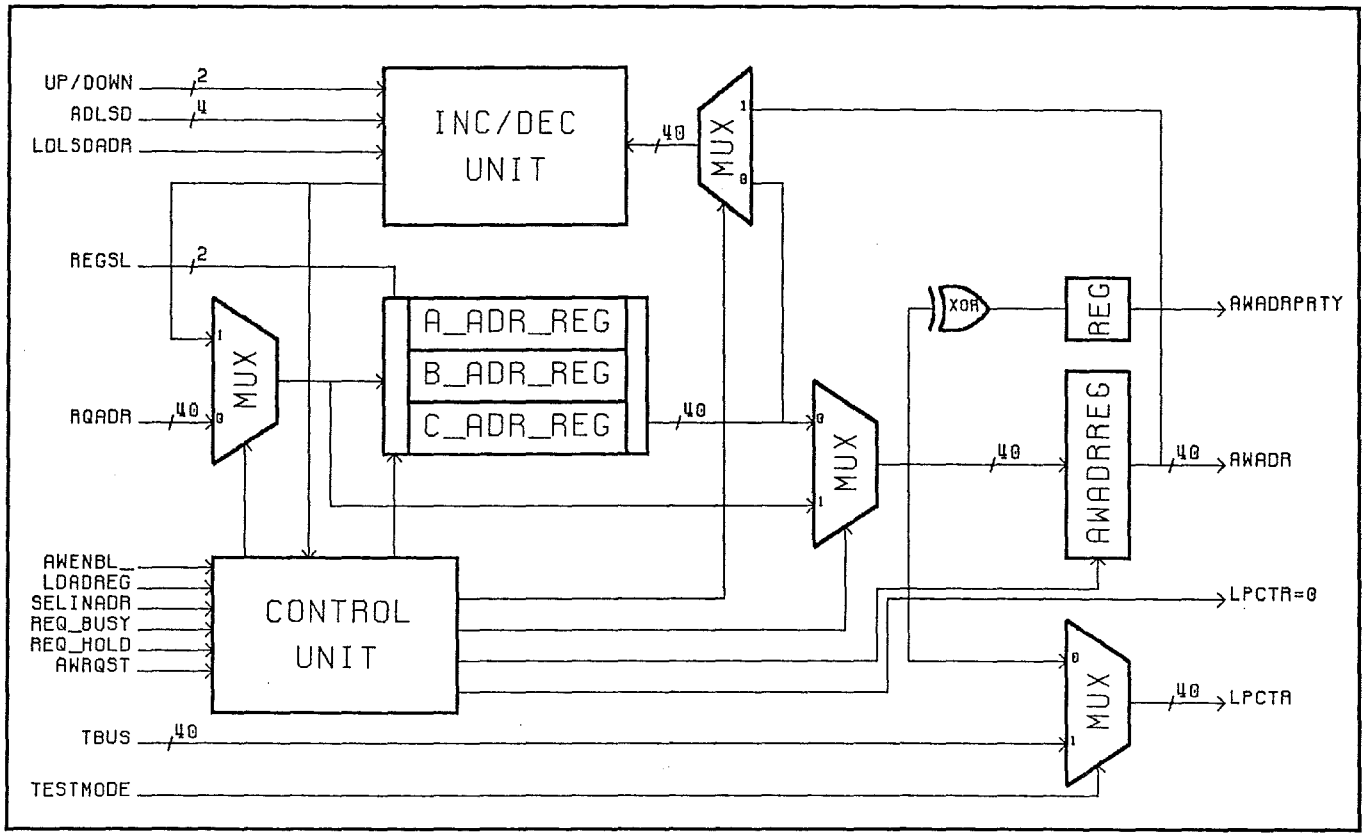


FIGURE 3-8 MEM REG ADDRESS MANIPULATOR UNIT



3.10.2 MEM REQ WRITE ALIGNMENT UNIT (REWAU)

The REWAU alions write Data to mod 10 boundaries and buffers data beyond the boundary for concatenation with the rext portion of write data. It can handle one write stream at a time. The end of a stream is recognized by a signal from RECTAG.

The REWAU consists of a Rotator/Corcatenator, a residue register (RREG), a Mem Data bus register with buffers and a parity generator (See Figure 3-9).

3.10.3 MEM REQ LENGTH MANIPULATOR (RELM)

The main function of this urit is to store the physical length of the operands and keep track of them. It has the capability of storing 3 10-bit lengths and decrementing them using a single digit arithmetic unit whenever such a command is issued by REGTAG. The other 3 main functions of RELM are to implement the loor counter functions along with maintaining the INC/DEC factor, to control the rotation of the write alignment unit and to implement the XM write algorithm via a mapperlike structure (See Figure 3-10).

3.10.4 MEM REQ CONTROL UNIT (RECTAG)

RECTAG controls the operation of the entire memory requestor. It also contains the tag generator unit which operates semi independently to generate the appropriate tags for reads and writes. The length status flag indicates the value of the lenoths stored in the three 10-bit wide registers in the logical lenoth unit. The number of outstanding reads/pops are indicated by a pair of lines which go to the control card (RBQFULL/RBQEMPTY). RECTAG along with RELM decides whether there needs to be a second cycle in case of write last commands. The general flow of commands in this unit is as such. (See Figure 3-11)

a. A command is received (Command Interface Unit).

b. It is either mapped to another command or passed.

c. It is decoded and control signals issued (Mapper, Mapper Support).

d. Some of the control signals form the AW_CMD. Other signals activate the tag generator unit as well as the rest of the requestor units.

3.10.4 MEM REQ CONTROL UNIT (RECTAG) (Continued)

TAG GENERATION UNIT

For every Read/Write there is a TAG associated with the command. The TAG Generator Unit facilitates this by driving the appropriate TAGs on the AW_TAG lines. The mechanism is as follows:

a. READS (To RBUSQ)

The unit receives a Read request from the RECTAG Mapper which increments the TAG and Request counters and simultaneously drives bits 0-2 of the AW-TAG lines with the previous contents of the TAG counter. (Bits 6-3 of AW_TAG will be 1100.)

b. READS (To OPERANDQ)

OPQ Request signal from RECTAGs Mapper will put the current XM page number on bits 1 and 2 of AW_TAG and bit 1 of REQ_SPADR on bit 0 of AW-TAG. The 2 counters are not affected. (Bits 6-3 of AW_TAG will be 1101.)

c. POPS (RBUSQ)

A RBQPOP signal from the queue control logic will decrement the request counter but not the TAG counter.

d. WRITES

Same as OPERANDQ Reads except that it is generated by a Write request signal and bits 6-3 of AW_TAG will be 1010.

REQFULL/REQEMPTY being or means that the request counter contains a value of 7/1. (See the following figure.)

FIGURE 3-9 MEM REQ WRITE ALIGNMENT UNIT

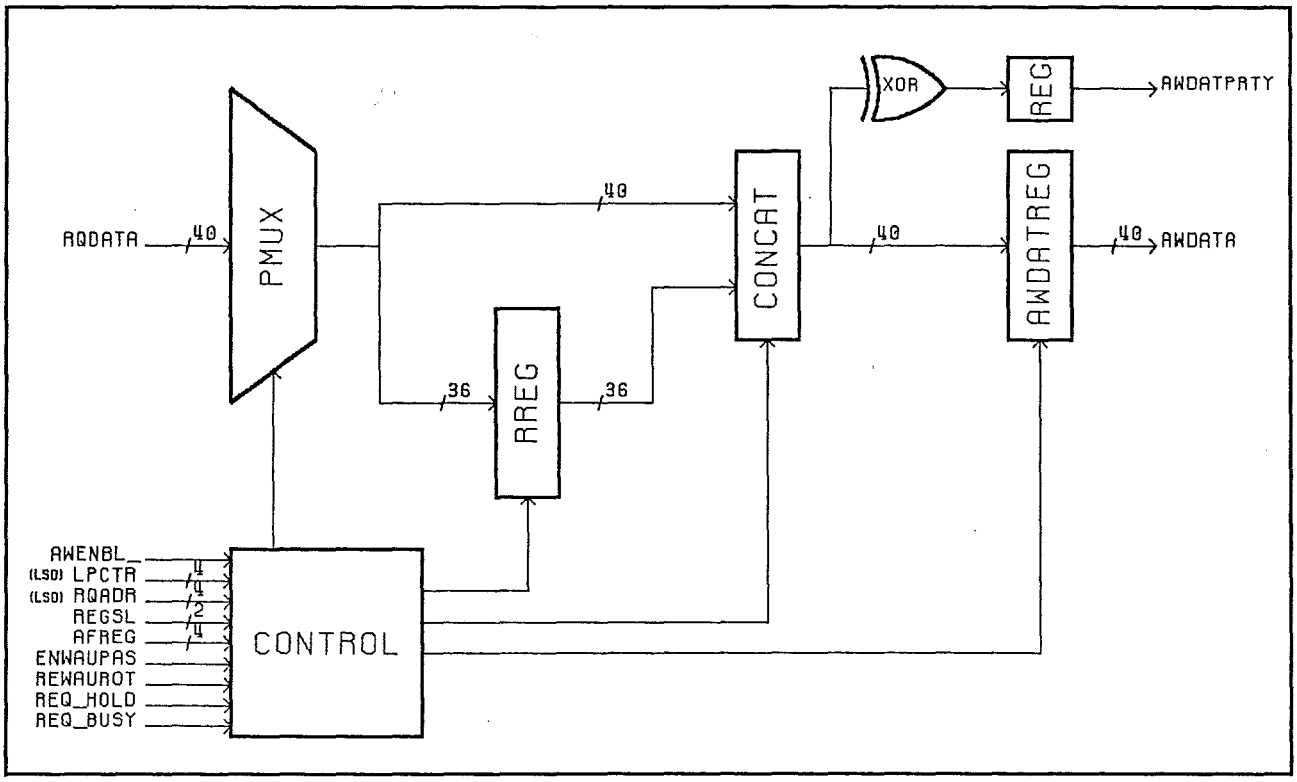


FIGURE 3-10 MEM REG LENGTH MANIPULATOR UNIT

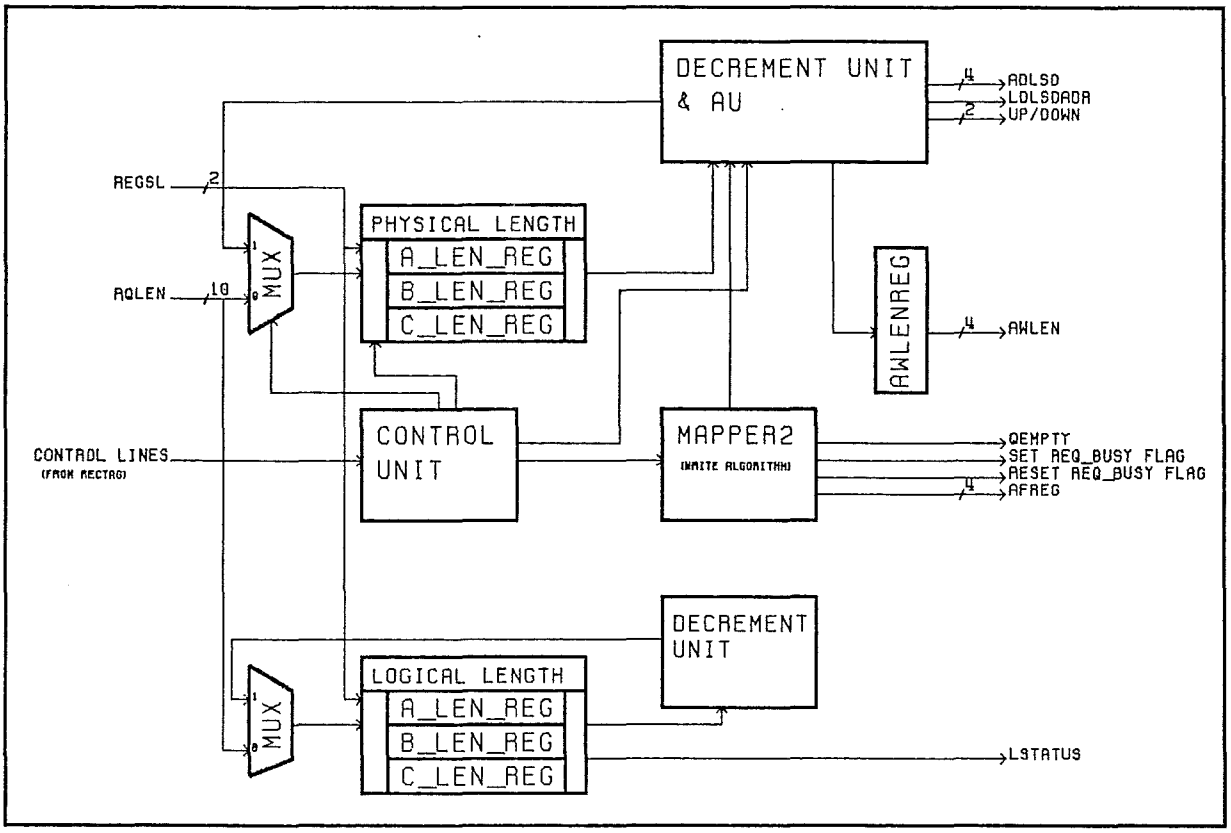


FIGURE 3-11 MEM REQ CONTROL AND TAG GENERATOR UNIT

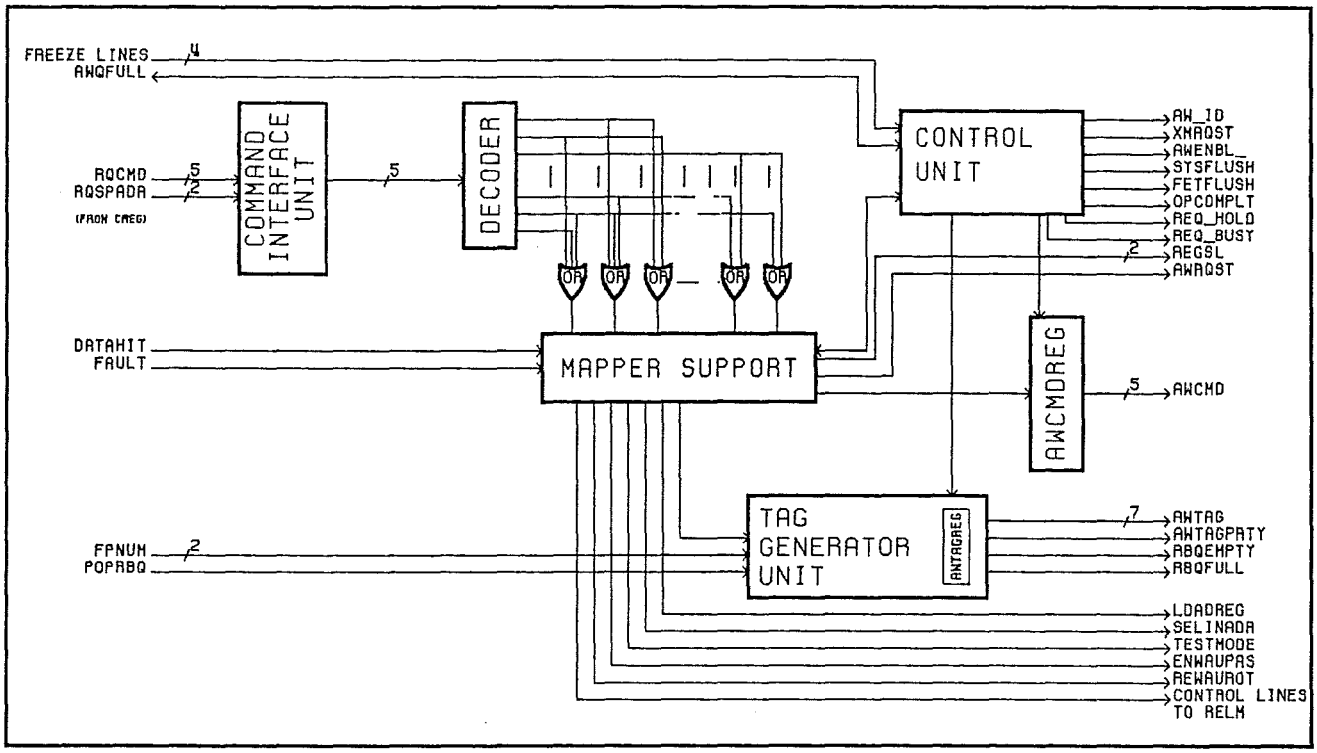
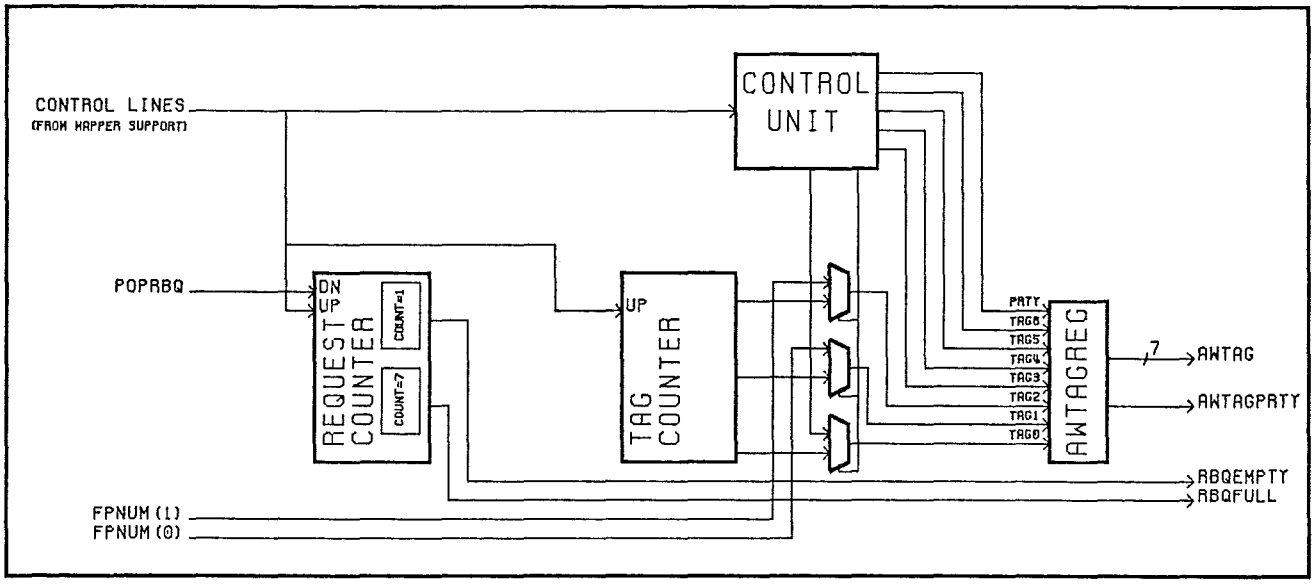


FIGURE 3-12 TAG GENERATOR UNIT



3.10.5 MEMORY REQUESTOR COMMANDS

A list of the commands associated with the Memory Requestor can be found in Section 4.6.10. However, a detailed description of those commands not found in the MCACM specification immediately follows.

A THRU command expects a length on the LSD of RQLEN ICU mux (0 --> 9) and an Address on the RQADR ICU lines. The remaining variations make use of the length and address registers in the requestor.

READ DATA (CMD=1)

This is a READ command that returns data back to the XM RBUSQ. The TAG counter in RECTAG decides the RBUSQ address.

UNCONDITIONAL READ OPERAND (CMD=2)

This is a READ command that returns data back to the XM OPERANDQ (A or B depending on the REQ_SPADR lines). This read always goes out no matter what the status of DATAHIT flag is.

CONDITIONAL READ OPERAND (CMD=2)

Conditional Read is used in an Optimal instruction flow to correct data in the operand queue in case of a Datahit indication from Fetch. It works without the need for microcode testing of Datahit in an attempt to save both microcode space and time.

During the first clocks of an optimal instruction a conditional read is fired to the addreses of the A and/or B operands. If a Datahit has occured, meaning that the data contained in the operand queue is invalid, a read is done on the address of the conditional read. The tags for the read are set to replace the appropriate entry in the OPQ.

If conditional read is issued and Datahit has not occured it is treated as a No Op, no request goes out to memory.

3.10.5 MEMORY REQUESTOR COMMANDS (Continued)

READ WITH LOCK (CMD=3)

This command is explained in the MCACM EDS. When using this command, microcode must make the appropriate alignment before sending data to memory. This command is only available in the THRU mode.

WRITE DATA (CMD=5)

This command writes the data on the RQDAT ICU lines to memory. The requestor will perform the necessary alignment on the data. A write command with the selected length < or = 10 will be converted into a write last command. Refer to Command 7.

WRITE LAST DATA AND SET OP COMPLETE (CMD=6)

Similar to WRITE DATA this command will perform a memory write and will also write the residual data during an independent write cycle, and set the op complete flag to Fetch. If during the independent cycle another requestor command is issued, XM will live freeze until the residual data is written to memory.

WRITE LAST DATA WITHOUT OP COMPLETE (CMD=7)

Identical to Command 6 except that op complete will not be set.

3.10.5 MEMORY REQUESTOR COMMANDS (Continued)

LOAD ADDRESS AND LENGTH REGISTERS (CMD=B)

This command will load the contents of the RQADR lines of ICU in the selected address register and the contents of the RQLEN lines of ICU in the corresponding length register. The status of the length information will be available next clock.

BRANCH TAKEN AND WRITE NEWPC (CMD=E)

This command will set the Fetch Flush line, put a code of 0C on the AW_CMD lines and send the NEWPC on the address lines.

BRANCH NTAKEN AND WRITE NEWPC (CMD=E)

Same as Command E except that a code of 0D will be put on the AW_CMD lines.

SYSTEM FLUSH AND WRITE NEWPC (CMD=12)

This command will set the SYS_FLUSH line, put a code of 0F on the AW_CMD lines and send the NEWPC to FETCH on the AW_ADDRESS lines.

INCREMENT ADR_REG BY LREG (CMD=13)

One of the loop counter functions, this command increments the selected ADR_REG by the inc/dec factor in LREG.

3.10.5 MEMORY REQUESTOR COMMANDS (Continued)

DECREMENT ADR_REG BY LREG (CMD=14)

Same as Command 13 except instead of incrementing the ADR_REG, it is decremented.

LOAD LREG (CMD=15)

Loads the loop counter inc/dec factor register. A value of 0 will be interpreted as 10.

LOAD LEN_REG (CMD=16)

This will load the data on the RQLEN 1CU lines into the selected length register. (Data is assumed to be 10 bits wide.)

SELECT LENGTH STATUS (CMD=17)

This command will set the L_STATUS flip flop if the content of the selected length register is less than or equal to 10, and will RESET L_STATUS if LENGTH > 10. If REQSPADR is 0, then the status of the latest selected length register will be preserved. L_STATUS is microcode branchable the clock after this command is issued.

OVERRIDE FAULT (CMD=18)

When there is a FAULT for the current instruction, all writes to memory will be inhibited. Override Fault will set the FAULTOVRIDE flag which will mask the Fault and let all subsequent memory writes to go thru. This flip flop will only be reset with a system Flush command.

3.10.5 MEMORY REQUESTOR COMMANDS (Continued)

OP COMPLETE WITHOUT WRITE (CMD=19)

This command will set the OP_COMPLETE line to Fetch.

LOAD ADR_REG (CMD=1A)

This will load the data on the RQADR ICU lines into the selected Address register.

ENABLE FORCE ERROR (CMD=1B)

This command forces errors in the selected arrays as follows:

RQLEN	ARRAY THAT RECEIVES THE ERROR
1	REAM (most sig slice)
2	REAM (least sig slice)
* 4	RELM
8	REWAU (most sig slice)
10	REWAU (least sig slice)

* Currently RELMs error detection circuitry is disabled.

SET TEST MODE AND DISABLE AW BUS (CMD=1C)

This command will set the TESTMODE flag which will disable the AWBUS drivers making some of the requestor signals and states visible to the data path. (Refer to Section 8.1 for more details.)

RESET TEST MODE AND ENABLE AWBUS (CMD=1D)

This will reset the TESTMODE flag and will remove the AWBUS disable signal.

3.10.6 XM WRITE ALGORITHM AND EXAMPLES

XM is the only processor module that can perform memory writes. All writes to memory are guaranteed to be on MOD 10 boundaries by the XM Memory Requestor. In order to accomplish this, the memory requestor implements the Write Algorithm.

XM's Write Algorithm has eight (8) sub-routines, two of them are used for WRITE commands whereas the remaining six are used for WRITE LAST commands.

Following is a detailed description of each of the sub-routines, which include conditions that invoke them. Following the description are some examples of types of writes.

DEFINITION OF TERMS USED:

- LSDADR = Least significant digit of the AWBUS address. (Digit 0)
- NLSDADR = Digit 1 of the AWBUS address.
- AWLEN = Actual data length on the AWBUS.
- PLENGTH = Physical length inside the Requestor Length Unit
- LLENGTH = Logical length inside the Requestor Length Unit.
- TENCOMP = 10 _ LSDADR (10's complement of LSDADR).
- AFREG = Alignment Factor register controls the rotation in the REWAU PMUX. This is the registered value of LSDADR.
- REQBUSY = Indicates the execution of the second write cycle.
- RESIDUE = Indicates the presence of valid data in RREG.
- RREG = Contains residual data in case of a write alignment.
- QEMPTY = Requestor Idle.

3.10.6 XM WRITE ALGORITHM AND EXAMPLES (Continued)

- OPCOMP = OP Complete.

- IRQCMD = Hypothetical register that stores the write command and register select in case of a second write cycle. (In actuality, it is derived from REQBUSY register).

Physical length indicates the actual number of digits remaining to be written.

Logical length indicates the number of digits microcode needs to write.

NOTE: In case of no write alignment, Physical and Logical lengths will be equal.

W R I T E A L G O R I T H M

WRITE PROCEDURE: (REQCMD = 5)

```
TENCOMP  <-  LSDADR
IF (LSDADR < > 0) AND (TENCOMP < PLENGTH) THEN:

     AFREG    <-  LSDADR              (FIRST VARIATION)
     AWLEN    <-  TENCOMP
     LSDADR   <-  0
     NLSDADR  <-  NLSDADR + 1
     PLENGTH  <-  PLENGTH - TENCOMP
     RESIDUE  <-  1


ELSE
   IF (LSDADR = 0) THEN:

     AWLEN    <-  0                   (SECOND VARIATION)
     NLSDADR  <-  NLSDADR + 1
     PLENGTH  <-  PLENGTH - 10


ELSE  (MICROCODE ERROR)
```

3.10.6 XM WRITE ALGORITHM AND EXAMPLES (Continued)

```
WRITE LAST PROCEDURE:  (REQCMD = 6, 7)

     TENCOMP  <-  10 _ LSDADR
     IF (LSDADR < > 0) AND (TENCOMP < PLENGTH) THEN:

          AWLEN    <-  TENCOMP              (1ST VARIATION)
          AFREG    <-  LSDADR
          LSDADR   <-  0
          NLSDADR  <-  NLSDADR + 1
          PLENGTH  <-  PLENGTH _ TENCOMP
          RESIDUE  <-  1
          IRQCMD   <-  RQCMD
          REQBUSY  <-  1


     ELSE
     IF (LSDADR < > 0) AND (TENCOMP > = PLENGTH) THEN:

          IF OPCOMP = 1 THEN OP COMPLETE.   (2ND VARIATION)
          AWLEN    <-  PLENGTH
          QEMPTY   <-  1
          RESIDUE  <-  0
          LSDADR   <-  0
          NLSDADR  <-  NLSDADR + 1
          PLENGTH  <-  0


     ELSE
     IF (LSDADR = 0) AND (REQBUSY = 0) AND (PLENGTH < = 10)
     THEN:

          IF OPCOMP = 1 THEN OP COMPLETE   (3RD VARIATION)
          QEMPTY   <-  1
          RESIDUE  <-  0
          AWLEN    <-  PLENGTH
          NLSDADR  <-  NLSDADR + 1
          PLENGTH  <-  0
          AFREG    <-  0
```

3.10.6 XM WRITE ALGORITHM AND EXAMPLES (Continued)

```
ELSE
IF (LSDADR =0) AND (REQBUSY = 0) AND (PLENGTH > 10)
THEN:

     REQBUSY  <-  RESIDUE                (4TH VARIATION)
     IF (REQBUSY = 1) THEN IRQCMD <- RQCMD
     ELSE     QEMPTY <-  1
     AWLEN    <-  0
     NLSDADR  <-  NLSDADR + 1
     PLENGTH  <-  PLENGTH - 10


ELSE
IF (LSDADR = 0) AND (REQBUSY = 1) AND (AFREG < PLENGTH)
THEN:

     AWLEN    <-  AFREG                 (5TH VARIATION)
     QEMPTY   <-  1
     LSDADR   <-  AFREG
     REQBUSY  <-  0
     IRQCMD   <-  0
     PLENGTH  <-  PLENGTH - AFREG
     RESIDUE  <-  0
     AFREG    <-  0


ELSE
IF (LSDADR = 0) AND (REQBUSY = 1) AND
(AFREG > = PLENGTH) THEN:

     IF (OPCOMP = 1) THEN OP COMPLETE   (6TH VARIATION)
     AWLEN    <-  PLENGTH
     QEMPTY   <-  1
     NLSDADR  <-  NLSDADR +1
     REQBUSY  <-  0
     IRQCMD   <-  0
     PLENGTH  <-  0
     RESIDUE  <-  0
     AFREG    <-  0
```

NOTE: Any write command (RQCMD = 5) with LLENGTH < = 10 will be converted to WRITE LAST (RQCMD = 7) inside the Requestor.

3.10.6 XM WRITE ALGORITHM AND EXAMPLES (Continued)

E X A M P L E # 1: SINGLE STREAM ADDRESS = 2004
LENGTH = 29

CLOCK #	RQCMD/ADR	AWADR	LLENGTH	PLENGTH	AWLEN	AWDATA
1	5/1	0	29	29	0	0
2	5/1	2004	19	23	6	XXXXDDDDDD
3	5/1	2010	9	13	0	RRRRDDDDDD
4	0/0	2020	0	3	0	RRRRDDDDDD
5	0/0	2030	0	0	3	RRRXXXXXXX

3.10.6 XM WRITE ALGORITHM AND EXAMPLES (Continued)

E X A M P L E # 2: DOUBLE STREAM

A Address = 1007
A Length = 28
B Address = 3921
B Length = 30

CLOCK #	RQCMD/ADR	AWADR	LLENGTH A/B	PLENGTH A/B	AWLEN	AWDATA
1	5/1	0	28/30	28/30	0	0
2	7/1	1007	18/30	25/30	3	XXXXXXXDDD
3	5/2	1010	8/30	15/30	0	RRRRRRRDDD
4	5/2	1020	8/30	8/30	7	RRRRRRRXXX
5	7/2	3921	8/20	8/21	9	XDDDDDDDDD
6	5/1	3930	8/10	8/11	0	RDDDDDDDDD
7	5/1	3940	8/10	8/10	1	RXXXXXXXXX
8	5/2	1027	0/10	5/10	3	XXXXXXXDDD
9	5/2	1030	0/10	0/10	5	RRRRRXXXXX
10	0/0	3941	0/0	0/1	9	XDDDDDDDDD
11	0/0	3950	0/0	0/0	1	RXXXXXXXXX

NOTE: During clocks 3, 6, and 8, XM will be forced into a live freeze state (AWBUSY) because of write alignment. Residual data is getting processed during those clocks.

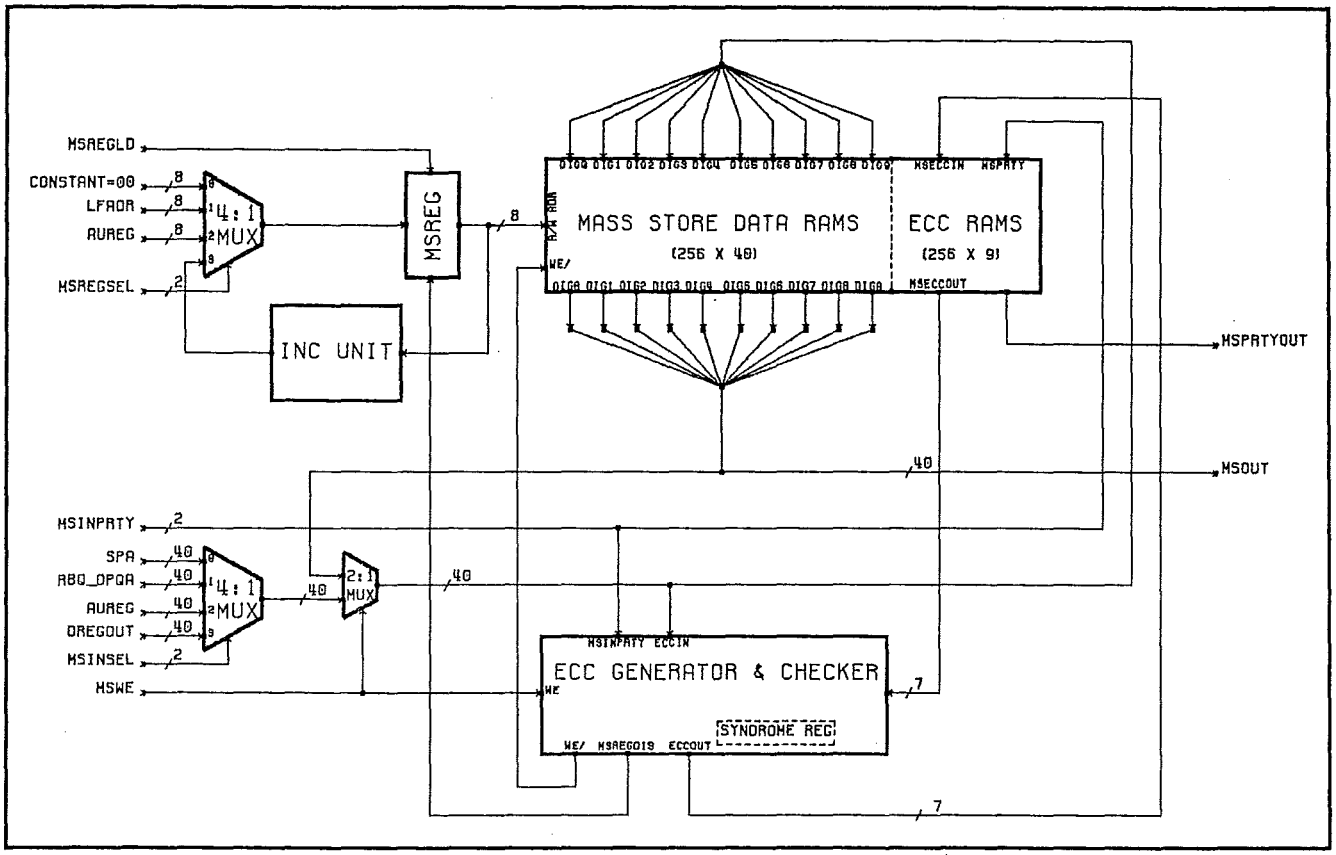
"D" in the data field stands for data coming from ICU.
"R" in the data field stands for data coming from RREG.

3.11 MASS STORE

The Mass Store Unit is an extension of scratchpad ram but has the restriction that it may be written or read during a clock but not both. It has only one output but has 256 words x 40 bit data storage capacity. ECC is stored with each entry to allow a microcode retry or mass store parity error (See Section 7.6). Figure 3-13 is a block diagram of this structure. Appendix D has a list of the ECC generation.

It is addressed via a register (MSREG) which is loadable from 2 data sources and has the increment by 1 capability; see Section 4.6.12 for details.

FIGURE 3-13 MASS STORE STRUCTURE



3.12 DREG

This is a 40 bit wide general purpose register that shares the ICU RQADR output with the memory requestor. Its output feeds the ICU. It is loaded when the 'DREGLD' bit in the CREG is set high.

3.13 CONTROL ENTRIES

The 'CNTLOUT' input to the ICU has the ten least significant bits sourced by the control section and the MSD by the FTREG. All other bits will be zero.

The following table indicates the addressing of the control out entries.

CTLOUTSEL	FFA_ADR	CNTLOUT
0	X	[ALEN-BLEN]
1	0	MCACM ERR VECTOR
1	1	ALEN
1	2	BLEN
1	3	CLEN
1	4	INTERRUPT INFO
1	5	CONSTANT=0/FREEZE CODE
1	6	INVALID
1	7	INVALID

Entry 5 can be used by microcode as a CONSTANT=0 during normal instruction execution. However, during function test, it contains the FREEZE CODE gererated as a result of a TEST CODE in the freeze control logic. (Refer to Appendix G for details.)

4 CONTROL SECTION

The Control Section provides sequence control for the XM, through the main sequence controller and several smaller interface controllers. It also provides some sequence tracing logic to aid in diagnostics.

The Control Section contains seven major parts (See Figure 4-1):

CRAM--control store ram 135 wide 16 K-word deep
CREG--control register 135 wide
Microinstuction address multiplexer (uImux)
Microprogram Counter (uPC)
Microcode Stack 14b wide 4 words deep
Condition Multiplexer(Conditmux)--2b wide
Fetch Queue 2--4 pages, 5x10 bit entries in each page

4.1 MICROINSTRUCTION MULTIPLEXER (uIMUX)

The Micro-instruction multiplexer is divided into two parts by the number of inputs required (See Figure 4-2).

The most significant 12b of instruction address may come from six sources:

1) Stack
2) uPC
3) Jump Address
4) PTREG
5) AC Controller
6) OpLit & Optimal Values

The final 2b of address comes from one source: The Test Condition Mux

The detailed combination of these inputs are defined in the description of the CRAM Address field of the CREG.

FIGURE 4-1 XM CONTROL STRUCTURE

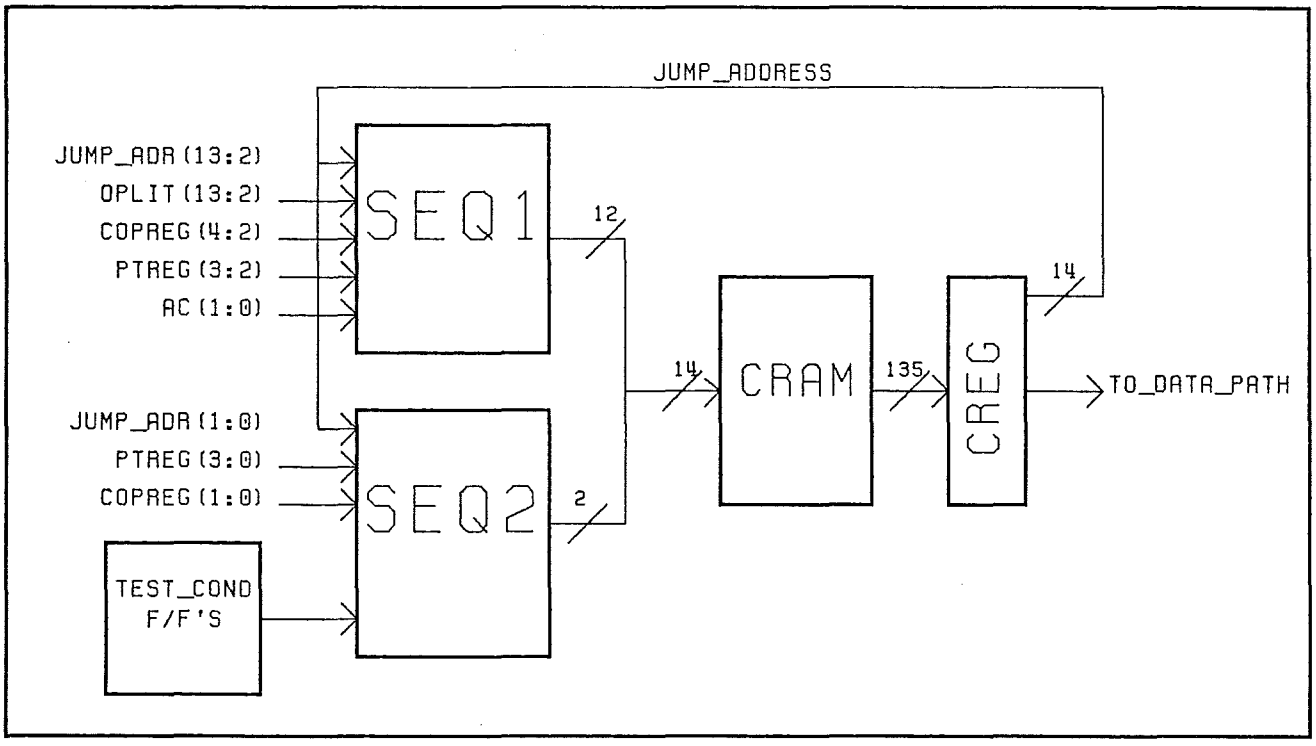


FIGURE 4-2 XM MICRO INSTRUCTION MUX

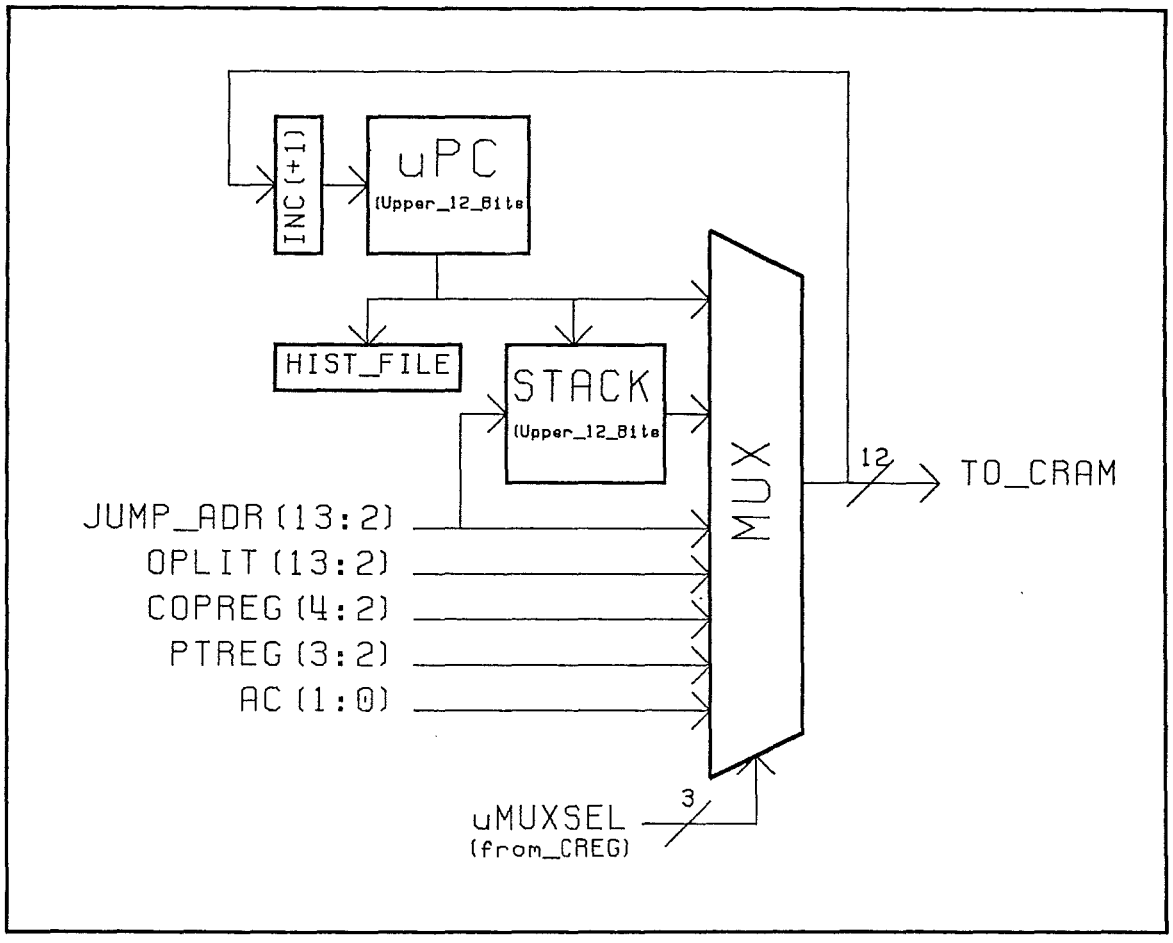
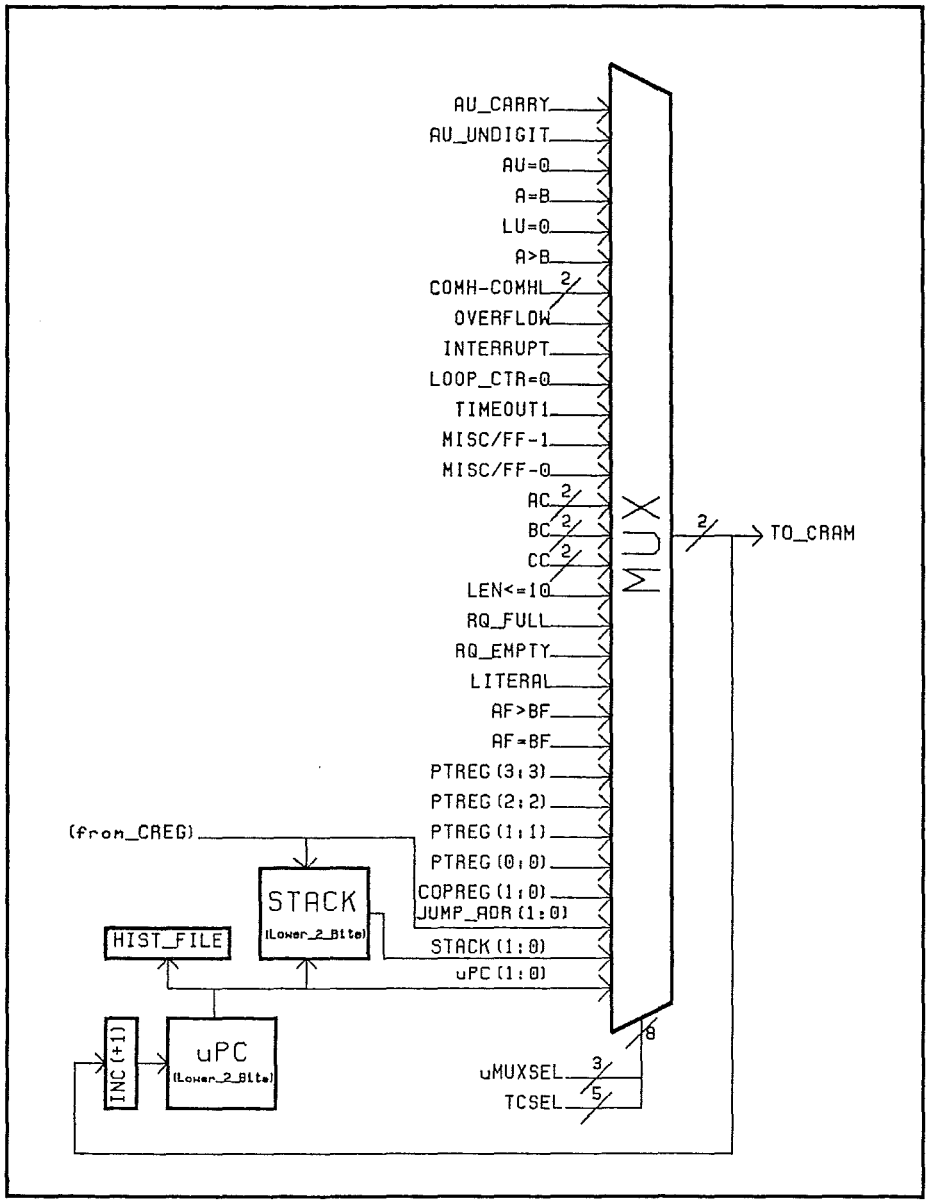


FIGURE 4-3 TEST CONDITION MULTIPLEXOR



4.2 CRAM AND CREG

The CRAM is 135b wide and 16K deep, composed of 16K x 1 ECL rams. It is a read only device to the XM, writing of data into the CRAM is done under maintenance control as described in Section 6.2.2.

The CREG is 135b wide and loads directly from the CRAM. It holds the u-instruction being executed and is used to hold information when loading the CRAM. The CREG's output feeds both the data and the control sections of the XM as well as a parity checker and the CRAM to load CREG data to the CRAM during the CRAM load cycle.

4.3 MICROCODE STACK

The Microcode Stack is 4 deep and 14b wide. It is a LIFO register file with a top of stack register. The stack is loaded from the uPC on a subroutine call (See PUSH CREG[14:14]), and popped on a return (See uMUXSEL : CREG[17:15=2]).

The stack is cleared by System Maintenance Clear, SYS_FLUSH, FETCH_FLUSH, an internal command, and by doing an OPLIT branch. When the stack is cleared, the register file and the top-of-stack register are untouched while the stack pointers are reinitialized.

The stack overflow and stack underflow are treated as errors. See Section 7.3.2. Associated with the stack is a "last branch" history file. (See Section 7.4)

4.4 TEST CONDITIONS MULTIPLEXER

The test condition multiplexer selects from the test condition flip-flops per the TCsel field of the uWord (See Section 4.6.11). The test conditions are appended to the next address information as specified by the CRAM address source field (See Section 4.6.11).

Several TCsel codes are as yet undefined, new test conditions may be added under the constraint that they must meet critical timing considerations.

The conditions presently being tested are:

Jmp[1:0]-------> Least significant 2 bits of uWord jump address field

UPC[1:0]-------> Least significant 2 bits of UPC+1

STACK[1:0]-----> Least significant 2 bits of STACK

PTREG[3:0]-----> PTEST register

AU carry-------> carry out of the MSD of the Arithmetic unit, for subtract it represents borrow-not

AU undigit-----> indicates the presence of an undigit on the AU inputs

AU=0-----------> test for Arithmetic unit output equal zero

A=B------------> Logic unit comparison "A equal to B"

LU=0-----------> Logic unit output equal to zero

A>B------------> "A greater than B" comparison result flag

4.4 TEST CONDITIONS MULTIPLEXER (Continued)

COM H, COM L---> together form the "global" comparison toggles for the XM, encoded as follows:

COM H	COM L	Meaning
0	0	comparison reset(null)
0	1	comparison low
1	0	comparison high
1	1	comparison equal

(See Section 4.4.1 for details on the Coms)

OVF----------> Overflow flag

INT----------> system interrupt line

Loop Cntr=0----> test for contents of loop counter equal to 0 or crossed 0 it reflects the counters conterts on the previous clock

TIMEOUT1-------> system timeout indicator

MISC1, MISC0---> Firmware programmable flags, see XM microcode EDS for their use.

AC----------> Address controllers
BC |
CC |

LEN<=10--------> Indicates the status of the selected logical length in the memory requestor

COPREG---------> The coprocessor register used to communicate between coprocessor and XMC (currently unused).

Number of
Extended
Syllable

RQFULL---------> When set, it indicates that the number of outstanding RBQ reads is equal to 7.

4.4 TEST CONDITIONS MULTIPLEXER (Continued)

RQEMPTY---------> When set, it indicates that the number of outstanding RBQ reads is equal to 1.

LIT FLAG--------> Indicates that the instruction has a literal.

AF>BF-----------> AF greater than BF flag from Fetch Queue 2.

AF=BF-----------> AF equal to BF flag from Fetch Queue 2.

NONFATAL--------> Indicates the occurrence of a nonfatal error inside the memory subsystem.

DHCODE (1:0)---> These two bits inform microcode of the status of conditional reads within the instruction page.

DHCODE	MEANING
0	No outstanding issue. NO OP
1	Contitional read B is needed.
2	Contitional read A is needed.
3	Contitional read A & B are needed.

4.4.1 COMS AND OVERFLOW (OVF)

The Comparison Flip-Flops (COMS) and the Overflow Flip-Flop (OVF) are pieces of the state of the machine which are passed from instruction to instruction. Instruction retry (See Section 7.7) may not be possible once state is changed unless a backup copy is maintained. For this reason both the coms and ovf are present in testable (global) and temporary form (local).

Microcode manipulates the local coms and OVF via the internal command field (See Section 4.6.13). Oplit branch loads the value of the local ffs into the global ffs without changing the local value, permitting the update of global coms as late as possible without using an extra clock.

If an update of the local coms is requested at Oplit branch, then both the local and global coms get updated.

4.5 FETCH QUEUE 2

Fetch Queue 2 is a separate portion of the Fetch-XM interface structure used to transfer testable conditions and data much shorter than 10 digits in length. It is 4 pages deep with 5 entries per page, its page select lines are common with Fetch queue 1 but entry selection is separate. For a diagram of Fetch queue 2 and related logic, see Figure 4-4.

4.5.1 OPLIT BRANCH

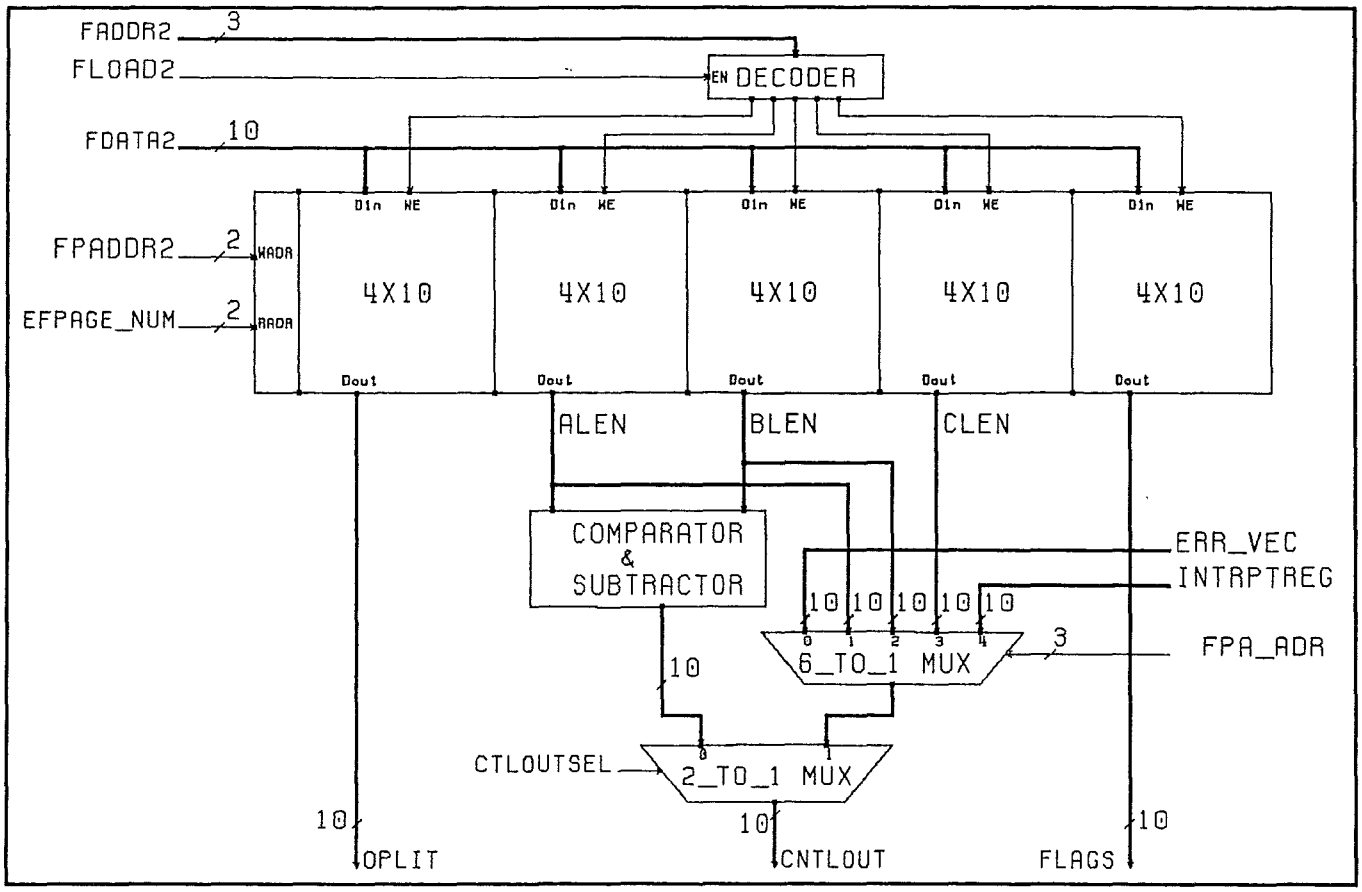
The Fetch module encodes the Opcode in a seven bit field (the Op vector), that is transferred to the XM, along with a bit indicating the presence of a literal. This eight bit code is stored in Fetch Queue 2 as the OPLIT entry. The concatenation of the OPLIT value and the Optimal flag creates a 512-way pointer into the first microword executed by the XM. There is an Optimal flag associated with each of the four fetch pages and its value is predetermined by the instruction. For a detailed description of this flag, refer to the XM microcode specification.

OPLIT branch refers to the clock which is executed to enter the first microword of each instruction. If there is no valid Fetch Page, while the microcode is trying to perform an OPLIT branch, then the XM will live freeze.

The OPLIT branch clock is one of the most active states in the hardware. Interrupts are checked only at OPLIT branch and if any get detected, the Interrupt handler in microcode is called. Once the branch is taken, the following registers get reset: stack pointers, misc. F/F's, NOTRY F/F, TIMEOUT1, DATAHIT, OPTIMAL, OPVALID, and Fetch Page Valid.

In addition, the global CONS and OVERFLOW get set based on the contents of their local copies. Finally, the signal EPOFFQUE$P is asserted to inform Fetch that the XM has released its current page.

FIGURE 4-4 FETCH QUEUE 2



4.6 XM MICROWORD DEFINITION

The following breaks down the microword into separate fields and then describes the coding of each.

4.6.1 SCRATCHPAD CONTROL

The following four fields control all read/write functions of the Scratchpad ram.

SP A ADDRESS CREG [126:123]

A four bit field that selects one of 16 locations to drive the A-port of the scratchpad.

SP B ADDRESS CREG [122:119]

A four bit field that selects one of 16 locations to drive the B-port of the scratchpad.

SP C ADDRESS CREG [114:111]

A four bit field that selects one of 16 locations to be written into if the SPWE is true.

WRITE ENABLE (SPWE) CREG [118:118]

Data selected by the SPIN mux in the ICU is written at the address specified by the C address field when this bit is set.

4.6.2 RBUSQ & OPQ CONTROL

The following three bits control RBQ/OPQ read requests.

RBOPQASEL CREG [127:127]

If this bit is a 1, the OPQA output is selected to drive the RBOPQA bus. If the bit is a 0, then RBUS is being accessed.

RBQOPQREQ CREG [41:41]

This bit indicates, either RBUS or OPQA is being accessed whenever it is set to a value of 1.

OPQBREQ CREG [129:129]

If OPQB is being accessed, this bit is high.

4.6.3 FETCH QUEUE ADDRESS - CREG [132:130]

A three bit field that selects one of 8 locations to drive the A-port of the Fetch queue from the current page. The B-port of the Fetch queue is wired to always select the B address of the instruction.

4.6.4 LIT FILE CONTROL

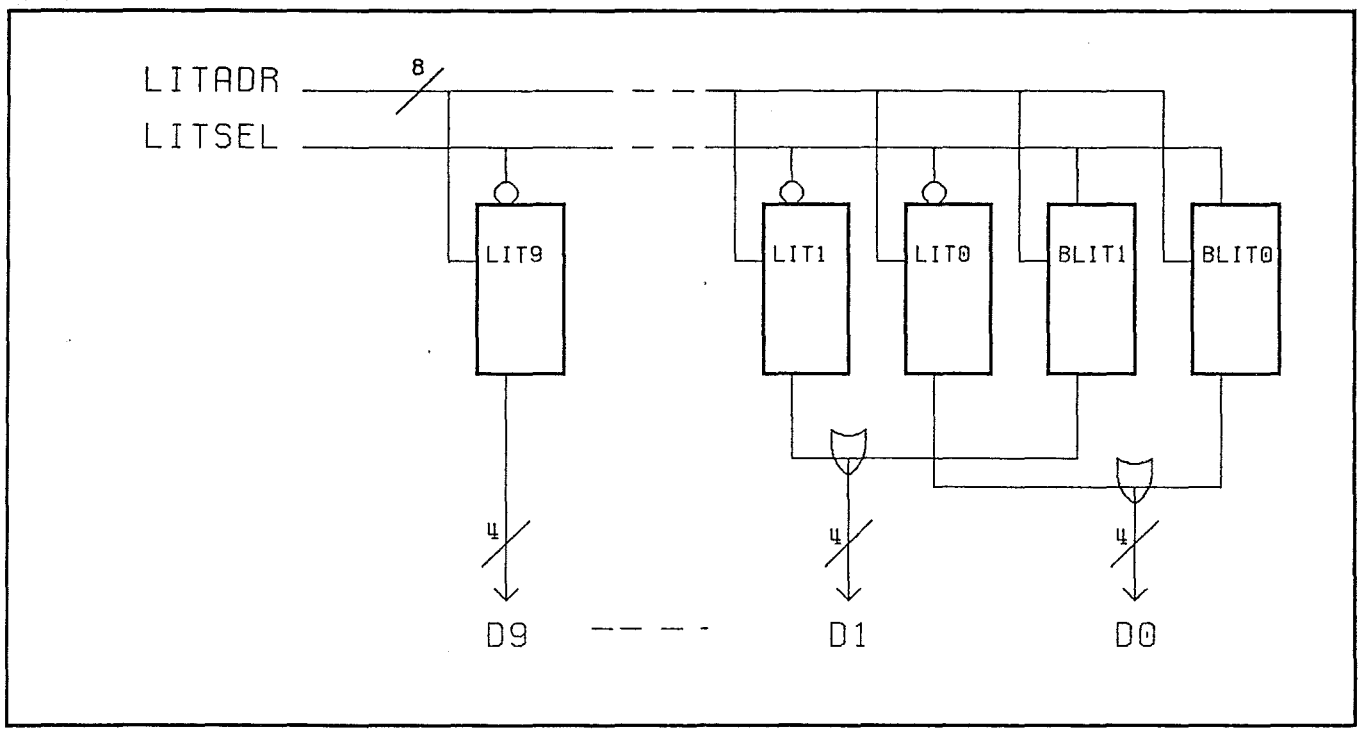
LITERAL SELECT - CREG [84:84]

Selects the byte literal instead of the litfile.

LIT FILE ADDRESS - CREG [92:85]

An eight bit field that selects the literal to be read. It also may be used to set up the mass store address. If the LITSEL bit is high, digits 9-2 will be forced to 0, and digits 1-0 will have a value - coming from the BLIT rams - equal to the litfile address. See Figure 4-5.

FIGURE 4-5 LITERAL FILE



4.6.5 ICU SELECT

Each of the outputs from the ICU is controlled by a distinct set of CREG bits as specified below. The Logic Unit and the Compare unit share the same input select mux within the ICU.

FIELD NAME	CREG	CODE	SOURCE
AUASEL	38:36	0	SPA
		1	FPA
		2	RPG/OPGA
		3	LITFILE
		4	MSOUT
		5	AUREG
		6	"0"
		7	DREGOUT
AUBSEL	35:33	0	SPB
		1	FPB
		2	OPGB
		3	LITFILE
		4	MSOUT
		5	AUREG
		6	CNTLOUT
		7	DREGOUT

4.6.5 ICU SELECT (Continued)

FIELD NAME	CREG	CODE	SOURCE
LUASEL	98:97	0	SFA
		1	FPA
		2	RBQ/QPQA
		3	LITFILE
LUBSEL	96:94	0	SPB
		1	FPB
		2	QPQB
		3	LFILE
		4	CNTLOUT
		5	DREGOUT
		6	AUREG
		7	'0'
PTSEL	110:108	0	SPA
		1	FPA
		2	RBQ/QPQA
		3	QPQB
		4	MSOUT
		5	AUREG
		6	N/A
		7	CTLOUT

4.6.5 ICU SELECT (Continued)

FIELD NAME	CREG	CODE	SOURCE
PMASEL	61:59	0	SPA
		1	FPA
		2	REQ/OPQA
		3	LITFILE
		4	AUREG
		5	DREGOUT
		6	CNTLOUT
		7	MSOUT
PMBSEL	58:57	0	SPB
		1	REQ/OPQA
		2	OPQB
		3	LITFILE
MSINSEL	82:81	0	SPA
		1	REQ/OPQA
		2	AUREG
		3	DREGOUT

4.6.5 ICU SELECT (Continued)

FIELD NAME	CREG	CODE	SOURCE
RQDSEL	70:68	0	SPA
		1	FPA
		2	FPB
		3	RBQ/OPQA
		4	LFILE
		5	MSOUT
		6	AUREG
		7	LU
RQASEL	67:65	0	SPA
		1	FPA
		2	FPB
		3	RBQ/OPQA
		4	LFILE
		5	MSOUT
		6	AUREG
		7	LU

4.6.5 ICU SELECT (Continued)

FIELD NAME	CREG	CODE	SOURCE
RQLSEL	64:62	0	SPP
		1	LITFILE
		2	MSOUT
		3	AUREG
		4	CNTLOUT
		5	"0"
		6-7	TBS
SPINSEL	117:115	0	SPA
		1	FPA
		2	RBQ/OPQA
		3	MSOUT
		4	AUREG
		5	LU
		6	LPCTR
		7	PNCNC

4.6.6 PMUX FUNCTION SELECT

The Pmux function is controlled through 3 fields, two control the reordering of digits within a word, the third controls the selection of the source of each output digit from one of the two reordered words.

PMUX A FUNCTION SELECT - CREG[56:54]

The defined functions of the PmuxA map is as follows:

	OUTPUT at digit										
Function	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Function Name
0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Pass
1	D0	D9	D8	D7	D6	D5	D4	D3	D2	D1	Rotate 1Right
2	D1	D0	D9	D8	D7	D6	D5	D4	D3	D2	Rotate 2Right
3	D2	D1	D0	D9	D8	D7	D6	D5	D4	D3	Rotate 3Right
4	D3	D2	D1	D0	D9	D8	D7	D6	D5	D4	Rotate 4Right
5	D4	D3	D2	D1	D0	D9	D8	D7	D6	D5	Rotate 5Right
6	D5	D4	D3	D2	D1	D0	D9	D8	D7	D6	Rotate 6Right
7	D6	D5	D4	D3	D2	D1	D0	D9	D8	D7	Rotate 7Right
8	D7	D6	D5	D4	D3	D2	D1	D0	D9	D8	Rotate 8Right
9	D8	D7	D6	D5	D4	D3	D2	D1	D0	D9	Rotate 9Right
0AH	D8	D6	D4	D2	D0	"0"	"0"	"0"	"0"	"0"	Del Zones left
0BH	"0"	"0"	"0"	"0"	"0"	D8	D6	D4	D2	D0	Del Zones Right
0CH	"F"	D9	"F"	D8	"F"	D7	"F"	D6	"F"	D5	Add Zones left
0DH	"F"	D4	"F"	D3	"F"	D2	"F"	D1	"F"	D0	Add Zones Right
0EH	D1	D0	D1	D0	D1	D0	D1	D0	D1	D0	BF Mask
0FH	***Function depends on (PMREG)***										SEL PMREG

With an appropriate concatenate, PMuxA=funct 0AH and PMuxB=funct 0BH, 20 digits of Alpha data can be zone stripped to 10 digits unsigned numeric in one clock.

PMUX B FUNCTION SELECT

The coding of Pmux B is identical to that of Pmux A. Whenever the PMREGLD bit is high, the PTREG is loaded into PMREGB.

4.6.6 PMUX FUNCTION SELECT (Continued)

CONCATENATE - CREG[48:44]

Concatenate provides the juxtaposition of the digit outputs of Pmux A and B. The field is coded as follows:

Function	Dig9	Dig8	Dig7	Dig6	Dig5	Dig4	Dig3	Dig2	Dig1	Dig0
0	PB9	PB8	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
1	PB9	PB8	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PA0
2	PB9	PB8	PB7	PB6	PB5	PB4	PB3	PB2	PA1	PA0
3	PB9	PB8	PB7	PB6	PB5	PB4	PB3	PA2	PA1	PA0
4	PB9	PB8	PB7	PB6	PB5	PB4	PA3	PA2	PA1	PA0
5	PB9	PB8	PB7	PB6	PB5	PA4	PA3	PA2	PA1	PA0
6	PB9	PB8	PB7	PB6	PA5	PA4	PA3	PA2	PA1	PA0
7	PB9	PB8	PB7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
8	PB9	PB8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
9	PB9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
0AH	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
0BH	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PB0
0CH	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PB1	PB0
0DH	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PB2	PB1	PB0
0EH	PA9	PA8	PA7	PA6	PA5	PA4	PB3	PB2	PB1	PB0

4.6.6 PMUX FUNCTION SELECT (Continued)

CONCATENATE - CREG[48:44] (Continued)

Function	Dig9	Dig8	Dig7	Dig6	Dig5	Dig4	Dig3	Dig2	Dig1	Dig0
0FH	PA9	PA8	PA7	PA6	PA5	PB4	PB3	PB2	PB1	PB0
10H	PA9	PA8	PA7	PA6	PB5	PB4	PB3	PB2	PB1	PB0
11H	PA9	PA8	PA7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
12H	PA9	PA8	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
13H	PA9	PB8	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
14	PA9	PB8	PB7	PB6	PB5	PB4	PB3	PA2	PA1	PA0
15	PB9	PA8	PA7	PA6	PA5	PA4	PA3	PB2	PB1	PB0
16-1F	*** To Be Specified ***									

**NOTE Abbreviations: PA(n) is content of the A part of Pmux, digit n; PB(n) is output of the B part of Pmux, digit n

4.6.7 ARITHMETIC FUNCTION SELECT CREG [32:29]

Selects the function of the arithmetic unit. Selections are as follows:

FIELD NAME	CREG	CODE	FUNCTION
AUFUNC	32:29	0	HOLD AUREG
		1	A+B Decimal
		2	A+B+1 Binary
		3	A+B+1 Decimal
		4	A+B Binary
		5	TBS
		6	A+B+CARRY Binary
		7	A+B+CARRY Decimal
		8	B-A Decimal
		9	B-A-1 Decimal
		A	A-B Binary
		B	A-B Decimal
		C	A-B-1 Binary
		D	A-B-1 Decimal
		E	A-B-Borrow/ Binary
		F	A-B-Borrow/ Decimal

4.6.8 LOGIC UNIT AND COMPARE

The Logic Unit and Compare are separate functions but use a common output of the ICU. Their control fields are coded as follows:

COMPARE FUNCTION CREG [101:1C1]

The Compare unit sets a pair of comparison flags: A > B and A = B

When the compare function bit is high comparison is done on only the MSD, otherwise a comparison is done across all 40 bits of the input data.

LOGIC FUNCTION SELECT - CREG[100:99]

Selection of the four possible functions of the Logic Unit is encoded in these two bits as follows:

CREG[100:99]	Function
0	A (AND) B
1	A (OR) B
2	A (XOR) B
3	NOT A (AND) B

4.6.9 PTEST

This field has two subfields which together control the function select and load enable for the Ptest register.

PTEST FUNCTION SELECT CREG[107:103]

Selects the Ptest function. It is used in conjunction with the PTREG load field (CREG[102:102]).

CREG[107:103]=0-15H Provides Ptest1 across all 40 bits of input, as received from ICU.
=16H-17H User programmable functions restricted to wire or combinations of a mapping of each input byte.
=18H-1FH Provides Ptest2 across the 4 bits of PTREG in addition to the 40 bits of input.

PTREG LOAD CREG[102:102]

Controls loading of PTREG, holding data when CREG[102:102]=0

4.6.10 MEMORY REQUESTOR CONTROL

Is a 7 bit field, divided into two separate fields, that controls the interface between XM and the rest of the processor.

ADDRESS REGISTER SELECT (REQ_SPADR) CREG[72:71]

This 2 bit field selects which Address/Length pair is to be used. Selection is as follows:

REQ_SPADR	MR register addressed
0	bypass registers, use ICU data
1	A_ADR_REG and A_LEN_REG
2	B_ADR_REG and B_LEN_REG
3	C_ADR_REG and C_LEN_REG

REQUEST COMMAND (RQCMD) CREG[77:73]

This five bit field selects the Memory Requestor action to be taken. The following is a list of presently defined functions:

RQCMD	RQREG	AWCMD	FUNCTION
00	0,1,2,3		NO OPERATION
01	0	01	READ DATA THRU
01	1	01	READ DATA AT A_ADR_REG
01	2	01	READ DATA AT B_ADR_REG
01	3	01	READ DATA AT C_ADR_REG
02	0	01	UNCONDITIONAL READ OPERAND A
02	1	01	CONDITIONAL READ OPERAND A
02	2	01	CONDITIONAL READ OPERAND B
02	3	01	UNCONDITIONAL READ OPERAND B
03	0	16	* READ THRU WITH LOCK (SPADR=1,2,3-->ERROR) DATA FIELD MUST HAVE DATA

4.6.10 MEMORY REQUESTOR CONTROL (Continued)

REQUEST COMMAND (RQCMD) CREG[77:73] (Continued)

RQCMD	RQREG	AWCMD	FUNCTION
04	0	1E	* READ DATA THRU UNCORRECTED (SPADR=1,2,3-->ERROR)
05	0	--	******ERROR******
05	1	1D	WRITE DATA AT A_ADR_REG
05	2	1D	WRITE DATA AT B_ADR_REG
05	3	1D	WRITE DATA AT C_ADR_REG
06	0	1D	WRITE DATA THRU AND SET OP COMPLETE
06	1	1D	WRITE DATA AT A_ADR_REG AND SET OP COMPLETE
06	2	1D	WRITE DATA AT B_ADR_REG AND SET OP COMPLETE
06	3	1D	WRITE DATA AT C_ADR_REG AND SET OP COMPLETE
07	0	1D	WRITE LAST DATA THRU WITHOUT OP COMPLETE
07	1	1D	WRITE LAST DATA AT A_ADR_REG WITHOUT OP COMPLETE
07	2	1D	WRITE LAST DATA AT B_ADR_REG WITHOUT OP COMPLETE
07	3	1D	WRITE LAST DATA AT C_ADR_REG WITHOUT OP COMPLETE
08	0	10	* WRITE PC
08	2	10	* WRITE PC
08	1	11	* READ PC
08	3	11	* READ PC
09	0	12	* READ ERROR ADDRESS ADDRESS FIELD MUST BE 0
09	2	12	* READ ERROR ADDRESS ADDRESS FIELD MUST BE 0
09	1	13	* READ ERROR REPORT ADDRESS FIELD MUST BE 0
09	3	13	* READ ERROR REPORT ADDRESS FIELD MUST BE 0

4.6.10 MEMORY REQUESTOR CONTROL (Continued)

REQUEST COMMAND (RQCMD) CREG[77:73] (Continued)

RQCMD	RQREG	AWCMD	FUNCTION
0A	0	06	* READ ECC CODE
0A	2	06	* READ ECC CODE
0A	1	07	* READ LIMIT TABLE ADDRESS FIELD MUST BE 0
0A	3	07	* READ LIMIT TABLE ADDRESS FIELD MUST BE 0
0B	0	--	*****ERROR*****
0B	1	--	* LOAD A ADR_REG & LEN_REG LENGTH STATUS WILL BE AVAILABLE NEXT CLOCK
0B	2	--	* LOAD B ADR_REG & LEN_REG LENGTH STATUS WILL BE AVAILABLE NEXT CLOCK
0B	3	--	* LOAD C ADR_REG & LEN_REG LENGTH STATUS WILL BE AVAILABLE NEXT CLOCK
0C	0	1A	* WRITE ECC CODE
0C	2	1A	* WRITE ECC CODE
0C	1	1B	* WRITE LIMIT TABLE
0C	3	1B	* WRITE LIMIT TABLE
0D	0	18	* WRITE BACK
0D	2	18	* WRITE BACK
0D	1	19	* WRITE BASE TABLE
0D	3	19	* WRITE BASE TABLE
0E	0	0C	BRANCH TAKEN & WRITE NEW PC ADDRESS FIELD HAS NEW PC
0E	2	0C	BRANCH TAKEN & WRITE NEW PC ADDRESS FIELD HAS NEW PC
0E	1	0D	BRANCH NTAKEN & WRITE NEW PC ADDRESS FIELD HAS NEW PC
0E	3	0D	BRANCH NTAKEN & WRITE NEW PC ADDRESS FIELD HAS NEW PC

4.6.10 MEMORY REQUESTOR CONTROL (Continued)

REQUEST COMMAND (RQCMD) CREG[77:73] (Continued)

RQCMD	RQREG	AKCMD	FUNCTION
0F	0	02	* WRITE I/O
0F	2	02	* WRITE I/O
0F	1	03	* READ I/O
0F	3	03	* READ I/O
10	0	05	* READ BASE TABLE (SPADR=1,2,3-->ERROR) ADDRESS FIELD MUST BE 0
11	0	04	* TEST ERROR DETECTION (SPADR=1,2,3-->ERROR)
12	0	0F	SYSTEM FLUSH & WRITE NEW PC ADDRESS FIELD HAS NEW PC (SPADR=1,2,3-->ERROR)
13	0	--	******ERROR******
13	1	--	INCREMENT A_ADR_REG BY LREG LOOP COUNTER FUNCTION
13	2	--	INCREMENT B_ADR_REG BY LREG LOOP COUNTER FUNCTION
13	3	--	INCREMENT C_ADR_REG BY LREG LOOP COUNTER FUNCTION
14	0	--	******ERROR******
14	1	--	DECREMENT A_ADR_REG BY LREG LOOP COUNTER FUNCTION
14	2	--	DECREMENT B_ADR_REG BY LREG LOOP COUNTER FUNCTION
14	3	--	DECREMENT C_ADR_REG BY LREG LOOP COUNTER FUNCTION
15	0,1,2,3	--	LOAD LREG LOOP COUNTER FUNCTION

4.6.10 MEMORY REQUESTOR CONTROL (Continued)

REQUEST COMMAND (RQCMD) CREG[77:73] (Continued)

RQCMD	RQREG	AWCMD	FUNCTION
16	0	--	******ERROR******
16	1	--	LOAD A_LEN_REG ONLY LENGTH STATUS WILL BE AVAILABLE NEXT LOCK
16	2	--	LOAD B_LEN_REG ONLY LENGTH STATUS WILL BE AVAILABLE NEXT LOCK
16	3	--	LOAD C_LEN_REG ONLY LENGTH STATUS WILL BE AVAILABLE NEXT LOCK
17	0	--	HOLD STATUS REGISTER CONTENT
17	1	--	SELECT A_LEN_REG ONLY STAT=0-->LENGTH > 10; STAT=1-->LENGTH < 10
17	2	--	SELECT B_LEN_REG ONLY STAT=0-->LENGTH > 10; STAT=1-->LENGTH < 10
17	3	--	SELECT C_LEN_REG ONLY STAT=0-->LENGTH > 10; STAT=1-->LENGTH < 10
18	0,1,2,3	--	OVERRIDE FAULT NOT TO BE USED DURING NORMAL INSTRUCTION EXECUTION
19	0,1,2,3	--	OP COMPLETE WITHOUT WRITE
1A	0	--	******ERROR******
1A	1	--	LOAD A_ADR_REG ONLY
1A	2	--	LOAD B_ADR_REG ONLY
1A	3	--	LOAD C_ADR_REG ONLY
1B	0,1,2,3	--	ENABLE FORCE ERROR
1C	0,1,2,3	--	SET TEST MODE & DISABLE AW BUS

4.6.10 MEMORY REQUESTOR CONTROL (Continued)

REQUEST COMMAND (RQCMD) CRFG[77:73] (Continued)

RQCMD	RQREG	AWCMD	FUNCTION
1D	0,1,2,3	--	RESET TEST MODE & ENABLE AW BUS
1E-1F	0,1,2,3	--	INVALID

NOTE: A. Descriptions of commands with an "*" are found in the MCACM specification.

B. Command 17 will also select the status of the address register that is being selected (LPCTR=0)

4.6.11 NEXT ADDRESS CONTROL

The following fields control sequencing for the XM, selecting branch address sources, microcode stack handling, and test conditions to be used in calculating the next address.

PUSH CREG[14:14]

Used in conjunction with CRAM Address (CREG[17:15]) to produce either a subroutine call or an unconditional PUSH of the Jump Address field into the micro-instruction stack. If the Next Address field (CREG[17:15]) is not a CONTINUE and this bit is "true", then the UPC + 1 is stored.

If the Next Address field is a CONTINUE and this bit is "true", then the value of the Jump Address field, from the CREG, is stored in the stack.

**NOTE: A "Push" on "Return" is not a valid operation and will produce undefined results.

MICROCODE JUMP ADDRESS CREG[13:0]

Provides a fourteen bit branch address for the microcode, parts of which are used as per CRAM Address field (CREG[17:15]).

TEST CONDITION SELECT (TCsel) CREG[28:24]

Selects the conditions used in CASE2, CASE4, CASE16 (for both controllers and FTREG).

TEST CONDITION SELECT (TCsel) CREG[28:24] (Continued)

SUMMARY OF MICROCODE TEST CONDITIONS

4-WAY TEST CONDITION, CASE4

Syntax	Code	Description
OPLIT	00H	512-Way OPLIT Branch
CONTINUE	01H	UPC + 1
RETURN	02H	Top of Stack
JMP	03H	JMP[1:2]
AC	04H	A address controller
BC	05H	B address controller
CC	06H	C address controller
F/F-10	07H	MISC F/F 10
COM-HL	08H	Global comparsion toggles
CMPA>=B	09H	CMP A>B & A=B
AU0&C	0AH	AU=0 & AU Carry
AF>=BF	0BH	AF>BF & AF=BF
FF1&A<=10	0CH	F/F1 & LEN <= 10
FF0&RQEMTY	0DH	F/F0 & No READ outstanding
PTR-31	0EH	PTREG-31
COPREG	0FH	Co-Processor Register
PTR-10	10H	PTREG-10
PTR-21	11H	PTREG-21
PTR-32	12H	PTREG-32
TBS&PTR-3	13H	TBS & PTR-3
A<=10&RQF	14H	LEN<=10 & 8 READ outstand
B<=10&RQF	14H	LEN<=10 & 8 READ outstand
C<=10&RQF	14H	LEN<=10 & 8 READ outstand
A<=10&RQEMTY	15H	LEN<=10 & No READ outstand
B<=10&RQEMTY	15H	LEN<=10 & No READ outstand
C<=10&RQEMTY	15H	LEN<=10 & No READ outstand
CARRY-UNDIGIT	16H	AU Carry & AU Undigit
PTR0&F/F0	17H	PTR-0 & MISC F/F-0
AUC&F/F1	18H	AU Carry & MISC F/F-1
PTR3&LU0	19H	PTR-3 & LU=0
EXTSYL	1AH	# of extended syllable
AU0&LIT	1BH	AU=0 & LIT
PTR0&LU0	1CH	PTR-0 & LU=0
A<=10&AU0	1DH	LEN<=10 & AU=0
B<=10&AU0	1DH	LEN<=10 & AU=0
C<=10&AU0	1DH	LEN<=10 & AU=0
DHCODE	1EH	Conditionel Read Status
	1FH	Undefined

TEST CONDITION SELECT (TCsel)-CREG[28:24] (Continued)

2-WAY TEST CONDITION

Syntax	Code	Description
AF=BF	00H	AF=BF
AF>BF	01H	AF>BF
FETCH-EVENT	02H	FETCH EVENT
TIMEOUT	03H	TIMEOUT 1
NONFATAL	04H	NONFATAL MCACM ERROR REPORTED
CTR<=10	05H	LEN<= 10
LPC0	06H	LOOP COUNTER = 0
	07H	Undefined
	08H	Undefined
	09H	Undefined
CARRY	0AH	CARRY bit on
BORROW/	0AH	CARRY bit on
	0BH	Undefined
CMPA>B	0CH	COMPARE A > B
CMPA=B	0DH	COMPARE A = B
OVF	0EH	OVERFLOW
INT	0FH	INTERRUPT
PTR-0	10H	PTREG 0
PTR-1	11H	PTREG 1
PTR-2	12H	PTREG
PTR-3	13H	PTREG 3
RQFUL	14H	& READ outstanding
RQEMTY	15H	NO READ outstanding
AU-UNDIGIT	16H	AU Undigit
F/F0	17H	MISC F/F-0
F/F1	18H	MISC F/F-1
LU0	19H	LU=0
EXTSYL	1AH	# of extended address
LIT	1BH	Literal flag
LU0	1CH	LU=0
AU0	1DH	AU=0
	1EH	Undefined
	1FH	Undefined

NOTE: Notice that for no. 0-F of case2 & case4, the test condition are completely differert. While for locations 10-1F, the lower bit of both case2 and case4 are the same.

4.6.11 NEXT ADDRESS CONTROL (Continued)

CRAM ADDRESS SOURCE CREG[17:15]

Selects the source of the next micro-instruction address in conjunction with TCselect (CREG[28:24]) as follows:

CREG[17:15]	Addr[13:8]	Addr[7:4]	Addr[3:2]	Addr[1:1]	Addr[0:0]
0	"0"/OLR	OLR	OLR	TC1	TC0
1	UPC	UPC	UPC	TC1	TC0
2	STK	STK	STK	TC1	TC0
3	JMP	JMP	COPREG	TC1	TC0
4	JMP	JMP	JMP	JMP	TC0
5	JMP	JMP	JMP	TC1	TC0
6	JMP	JMP	PTRG	TC1	TC0
7	JMP	JMP	AC	TC1	TC0

**NOTE: Abbreviations:
PC =MicroPC
TC0,TC1=Test Conditions as selected by TCsel (CREG[28:24])
Jmp =Microcode jump address (CREG[13:6])
Stk =Microcode Stack
Ptrg =Ptest register
OLR =OpLit register
AC =A address controller
COPREG =Co-Processor Interface Register

OpLit branch bumps Fetch page, sets Fetch page empty clears the stack, and on Fetch page not full causes a freeze condition. This branch is only rational once per instruction as the last clock of the instruction.

4.6.12 MASS STORE CONTROL

MS WRITE ENABLE CREG [83:83]

Mass Store will be written when MS write enable is "1"; otherwise, it will be read.

MSREG LOAD CREG [80:80]

The MSREG will be loaded when MSREG Load is "1"; otherwise, its contents will be held.

MSREG INPUT SELECT CREG [79:78]

MSREG may be loaded in four different ways from the data section. Details are as follows:

CREG[79:78]	MSREG source
0	Fixed constant "00"
1	Littile Address
2	AUREG
3	MSREG+1

4.6.13 INTERNAL COMMAND FIELD CREG[23:18]

Provides for the control of miscellareous flags in the XM, encoded as follows:

Internal Command Field - CREG[23:18]
Total # of commands available: 64
Total # of commands defined: 56

00 Null
01 Clear Error Vector
02 Load Task # Compare
03 Load Measurement register
04 LIX Ok - registered
05 Branch Ok - registered
06 Set IPC Interrupt - registered
07 Set XM Event - registered
08 Set Internal F Bus enable (Maint only)
09 Clear Internal F Bus enable (Maint only)
0A Clear Fetch Event
0B Set NoTry
0C not implemented
0D Load TEST CODE in freeze control unit (Maint only)
0E Reset TEST CODE in freeze control unit (Maint only)
0F " "
10 Read Task Timer
11 Clear Stack
12 Reserved
13 Test timer counters enable
14 Load Task Timer
15 Reserved
16 Reserved
17 Clear Time Out counter
18 Set Force Error flip-flop
19 Spare sourced by INTRPT array #1
1A Internal RBus Enable
1B Reset SMC Interrupt
1C Clear Force Error flip-flop
1D Reserved
1E Spare
1F Spare
20 Set Ovf low
21 Set Ovf high

4.6.13 INTERNAL COMMAND FIELD - CREG[23:18] (Continued)

22 Clear Misc0
23 Set Misc0
24 Clear Misc1
25 Set Misc1
26 Clear Misc0/1
27 Set Misc0/1
28 Set Coms equal
29 Set Coms high
2A Set Coms low
2B Clear Coms
2C Set Coms on A>=B
2D Set Coms on PTREG-1:0
2E Set Coms on LU=0
2F Set Coms on A>B & AU=0
30 Load Interrupt Register A 7 bits
31 Load Interrupt Register B 7 bits
32 Load Interrupt Mask 7 bits
33 Psuedo POP FETCH PAGE
34 Load Mode Register 8 bits
35 reserved
36 Load Miscellaneous Register 8 bits
37 reserved
38 Read Mode Register
39 Read Miscellaneous Register
3A reserved
3B reserved
3C Read Interrupt Register A
3D Read Interrupt Register B
3E Read Interrupt Mask
3F Read Test Conditions 6 bits - Global Coms & Ovf, Misc, Fetch event

Notes:

Set IPC Interrupt - causes the IPC flags in all other XM's to set. It is inhibited by OFFLINE. The output is driven from a register.

Clear Error Vector - clears the MCACM error vector in the RBUS interface.

4.6.13 INTERNAL COMMAND FIELD - CREG[23:18] (Continued)

Set XM Event - sets the XM Event flag in Fetch. The output is driven from a register.

Load Measurement register - loads the Measurement register on the data card.

Lix Ok - indicates to Fetch that the XM has just written the index registers. The output is driven from a register.

Branch Ok - indicates to Fetch that a branch was correctly predicted. The output is driven from a register.

Clear Stack - along with POP FETCH PAGE, Fetch Flush or System Flush generates a signal used to clear the microcode's stack.

Set & Clear Bus load - controls a register that causes the current fetch page to be loaded from Fetch or an internal data path.

Psuedo POP FETCH PAGE - generates a POP FETCH PAGE command within the INTRPT array. Execution of this command will update the COMS and OVF, set the Trace/Debug Interrupt if the Trace or Debug Mode bits are set, issue a Clear Stack, clear NoTry, Misc0 & 1, Timeout1, and the Coprocessor Data Valid flag. The OREG PARITY WILL BE CHECKED when this command is executed so it would be test if it were all zeros.

Set Coms on A>=B.

	A>B	A=B
High	1	0
Equal	0	1
Low	0	0

Set Coms on PTREG=10. Comh = PTR-1
Coml = PTR-0

Set Coms on LU=0. Equal if LU=0
High if LU<>0

Set Coms on A>B & AU=0. Equal if AU=0
High if AU><0 & A>B
Low if AU><0 & NOT(A>B)

4.7 XM CREG DEFINITIONS SUMMARY

	NAME	CREG	FUNCTION
*	SPARE	134:134	Used to load the SPARE RAMS
	PARITY	133:133	Parity bit
	FPAADR	132:130	FPAGE A-port address
	OPQBREQ	129:129	OPQB request if = 1
*	SPARE	128:128	
	RBOPQSEL	127:127	RBUSQ/OPQA port select
	SPAADR	126:123	Scratchpad A-port address
	SPBADR	122:119	Scratchpad B-port address
	SPWE	118:118	Scratchpad write enable
	SPINSEL	117:115	Scratchpad input select
	SPCADR	114:111	Scratchpad write address
	PTSEL	110:108	PTEST input select
	PTFUNC	107:103	PTEST function select
	PTREGLD	102:102	PTREG LOAD enable
	CMPFUNC	101:101	MSD compare enable
	LUFUNC	100:99	LU function select
	LUASEL	98:97	LU A-input select
	LUBSEL	96:94	LU B-input select
*	SPARE	93:93	
	LFILEADR	92:85	LFILE address field
	LITSEL	84:84	selects byte literal from CREG
	MSWE	83:83	Mass Store write enable
	MSINSEL	82:81	MSTORE input select
	MSREGLD	80:80	MSTORE address register load
	MSADRSEL	79:78	MSTORE address register input select
	REQCMD	77:73	MEMORY REQUESTOR command
	REQSFADR	72:71	MEMORY REQUESTOR register select
	RQDSEL	70:68	WRITE Data Queue input select
	RQASEL	67:65	MEMRQSTR ADDR input select
	RQLSEL	64:62	MEMRQSTR LEN input select
	PMASEL	61:59	PMUX A input select
	PMBSEL	58:57	PMUX B input select
	PMAFUNC	56:53	PMUX A function select
	PMBFUNC	52:49	PMUX B function select
	CNCATFUNC	48:44	Concatinate function select
	PMREGLD	43:43	PMUX function field source select
	CTLOUTSEL	42:42	Control output bus select
	RBOPQREQ	41:41	RBUS/OPQ request line
	DREGLD	40:40	DREG load enable
*	SPARE	39:39	

4.7 XM CREG DEFINITIONS SUMMARY (Continued)

NAME	CREG	FUNCTION
AUASEL	38:36	AU A-input select
AUBSEL	35:33	AU B-input select
AUFUNC	32:29	AU function select
TCONDSEL	28:24	TEST CONDITIONS select
INTCMD	23:18	Internal commard
UMUXSEL	17:15	NEXT microinstructior select
PUSH	14:14	write erable tc STACK
JUMPADR	13:0	Branch address field

4.8 TIMER FUNCTIONAL DESCRIPTION

The Timer array provides the Task Timer, a timeout counter, and the decode of 32 bits of the Internal Command.

4.8.1 TASK TIMER

The Task Timer is an eight digit BCD counter that decrements every 10 microseconds. It is loaded by internal command 14 with the 8 msd's of the DREG. It is placed on the FFB data path by internal command 10. When the most significant digit becomes zero the Timer Interrupt signal is generated. If all eight digits become zero the Timer Fault is generated. The Timer will go on to 99999999 and continue counting down. The Timer Interrupt and Fault are masked at the Interrupt array. The corresponding mask bit must be set to enable the reporting of the interrupt. See Section 4.9.1 INTERRUPT REGISTERS. After maintenance clear both the fault and the interrupt will be disabled. The array receives a one microsecond square wave from the IOMC. This signal is synchronized with the system clock and divided by ten within the Timer array for use by the Task Timer. The divider is reset when the Task Timer is loaded. Internal command 13 causes the divider to decrement on every system clock rather than the one microsecond clock. This function is meant for use only by test routines.

4.8.2 TIMEOUT COUNTER

The Timeout counter is a three bit gray coce counter that is enabled by the 0.2 or 0.4 second output of the Task Timer. This gives a timeout of 1.6 or 3.2 seconds. The value of the timeout is selected by an external strap. The timeout counter is reset at the start of each instruction, upon entry to the interrupt microcode by the OpLit and interrupt signals or by the clear timeout counter internal command (the Task Timer must be cleared seperately). Because the timeout is driven by the Task Timer, which is not reset at the start of each instruction, the timeout will vary between 1.4 and 1.6 or 2.8 and 3.2 seconds. Logic in the Interrupt array generates the Timeout1 and Timeout2 signals from the timeout generated by the Timeout counter. The Timeout1 and Timeout2 signals will have mask bits. The Timeout will be enabled after a maintenance clear.

4.8.3 INTERNAL COMMAND DECODER

The portion of the Internal Command Decoder in the Timer Array decodes the Timer Load, Read and Clear commands, the Force Error Set and Clear commands, and the Timer Clear and Test Commands. The Set Force Error goes to a flip-flop which generates the Forcerr signal for the rest of module. Forcerr also goes to a flip-flop so that ModuleOK will be active at the same time as the error generated by the Forcerr. The "Check the checkers" microcode will have to set Forcerr and generate an error condition on every clock until Forcerr is reset.

The Timer Test command causes the prescaler to the Task Timer to increment on every clock rather than every microsecond. The Timer Clear command only clears the three bit timeout counter.

4.9 STATE REGISTERS

4.9.1 INTERRUPT REGISTER

The Interrupt Register consists of two 6-bit interrupt registers, A and B, and a 6-bit mask register. Along with the Interrupt conditions, the register also holds the Timeout 1 and 2 bits. The interrupt bits are OR'd into a single interrupt register which drives the XM sequencer logic. The interrupt conditions are set by hardware events. They are cleared by an Internal Command, Pop Fetch Page in the case of Timeout 1 or the SMC in the case of Timeout 2. Internal commands (ICMD) allow all three of the registers to be loaded from the 7 least significant bits of the DREG or read via the CONTROL OUT bus.

INTERRUPT REGISTER A (MASKABLE)

BIT	DEFINITION	SET BY	CLEARED BY	SENSED BY
0	NORMAL IO	IOMC	INTCMD	INTRPT
1	IO ERROR	IOMC	INTCMD	INTRPT
2	REAL TIME IO	IOMC	INTCMD	INTRPT
3	TSK TIMR INT	TIMER	INTCMD	INTRPT
4	TSK TIMR FLT	TIMER	INTCMD	INTRPT
5	TIMEOUT 1	TIMER	POP_FP	TCOND

INTERRUPT REGISTER B (NON-MASKABLE)

BIT	DEFINITION	SET BY	CLEARED BY	SENSED BY
0	TIMEOUT 2	TIMER	CHAIN	DFREEZE
1	SMC INTRPT	STOP	INTCMD	INTRPT
2	FAULT	FNCTL	INTCMD	INTRPT
3	TRACE/DEBUG	MODE	INTCMD	INTRPT
4	IPC	ALL XM's	INTCMD	INTRPT
5	SNAP PIC TAKN	CHAIN	INTCMD	INTRPT

Explanation of terms used in the above tables:

INTCMD ---> XM Internal Command field from CREG
POP_FP ---> Oplit Branch
TCOND ---> Test Condition MUX
MODE ---> MODE register

4.9.2 MODE REGISTER

The Mode register contains the 8 bits listed below. They are read and written by Internal Command. Snap Enable drives a line to the SMC. The assignment of the other bits is arbritary.

Mode Indicator Register

0 Snap Enable
1 Trace Enable
2 Privaledged
3 Soft Fault Enable
4 Debug Enable
5 State Mode 0
6 State Mode 1
7 State Mode 2

4.9.3 TEST CONDITION REGISTER

The Test Condition register contains the six bits listed below. They are set and cleared by Internal Command except for FETCH EVENT which is set by the FETCH module. The bits can be branched on by the XM sequencer. They can also be read out via the CONTROL OUT bus by Internal Command.

Test Conditions

0 ComH
1 ComL
2 OVF
3 Misc0
4 Misc1
5 Fetch Event

4.9.4 MISCELLANEOUS REGISTER

The Miscellaneous register contains the following 7 bits. They can be set or cleared by Internal Command via the DREG or by some other hardware event as listed. They are also sensed by other hardware as listed and can be read by Internal Command via the CONTROL_OUT bus.

Miscellaneous	Set by	Cleared by	Sensed by
0 Literal flag	POP_FP		XM sequencer
1 MOPOK	XM ICMD	XM ICMD	Back Plane
2 ProHalt	"	"	SMC
3 Software Error Snap	"	chain	SMC
4 NOTRY	"	POP_FP	Dead Freeze
5 Fetch Flush	ext	Follows input	XM, FETCH
6 System Flush	ext	Follows input	XM, FETCH, MCACM

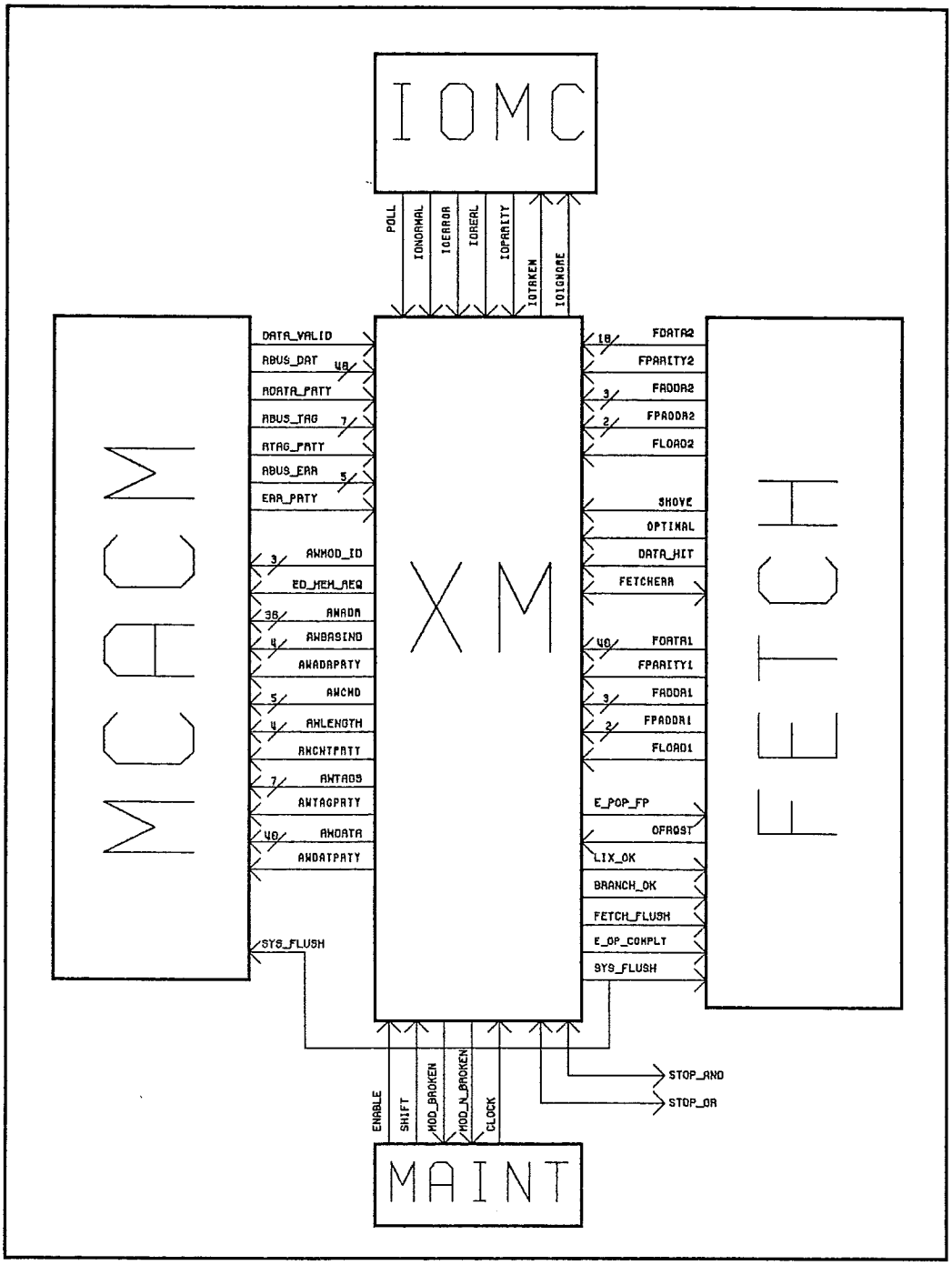
5 XM INTERMODULAR INTERFACES

Communication between the XM and the Fetch Module or the Memory Controller is via a combination of the following busses:

FBUS	Section 5.3
AWBUS	Section 5.4
RBUS	Section 5.5

See Figure 5-1.

FIGURE 5-1 XM INTERMODULAR INTERFACES



5.1 PHYSICAL/ELECTRICAL CONNECTION

All bus signals are active Hi (i.e. toward ground). Levels are ECL compatible.

The FBUS is comprised of 70 bussed lines. The AWBUS has 105 bussed lines. The RBUS has 56 bussed lines. There are a total of 215 bus-related backplane signals. See Section 5.2.

For backplane pinout assignments of signals on both boards of the XM, see Appendix B.

5.2 EXECUTE MODULE "XM" BACKPLANE LOGIC SIGNALS

FIGURE 5-2 XMC BACKPLANE LOGIC SIGNALS

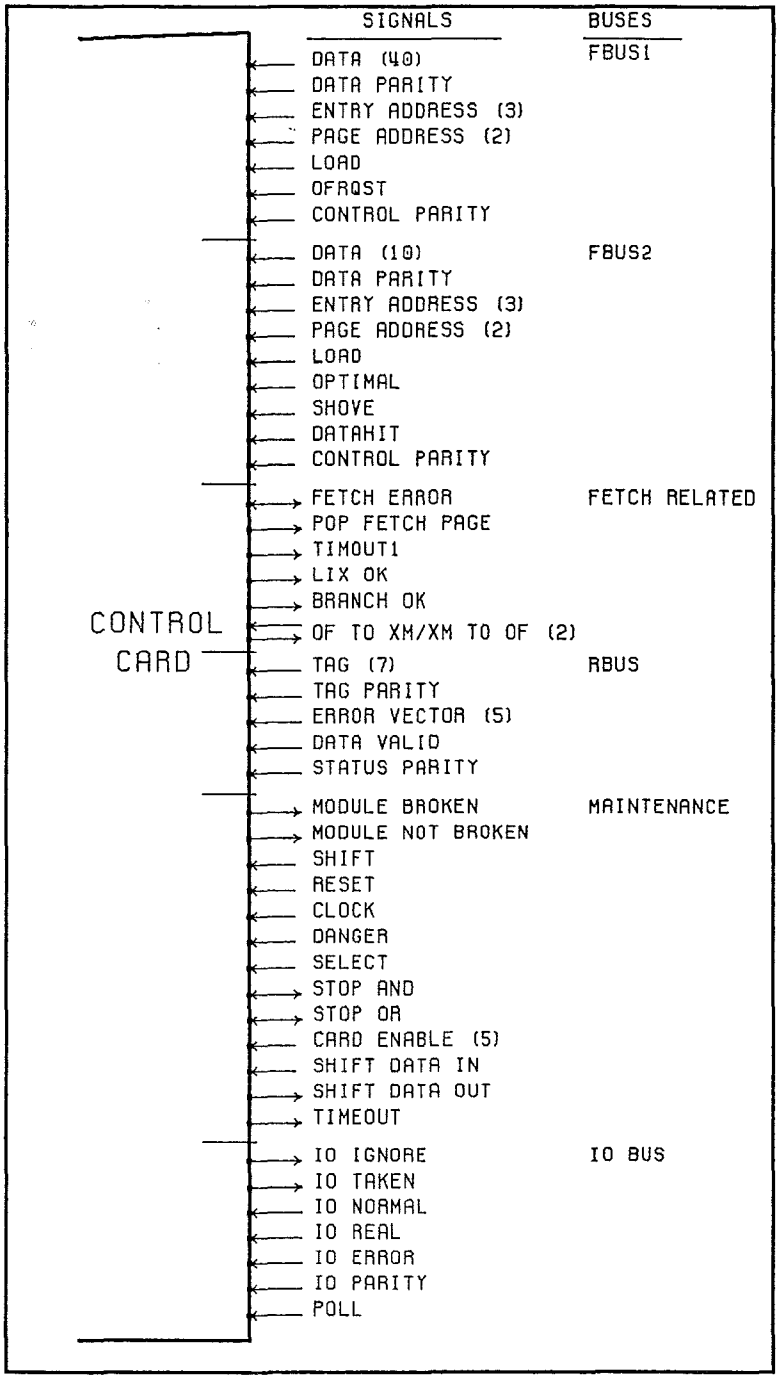
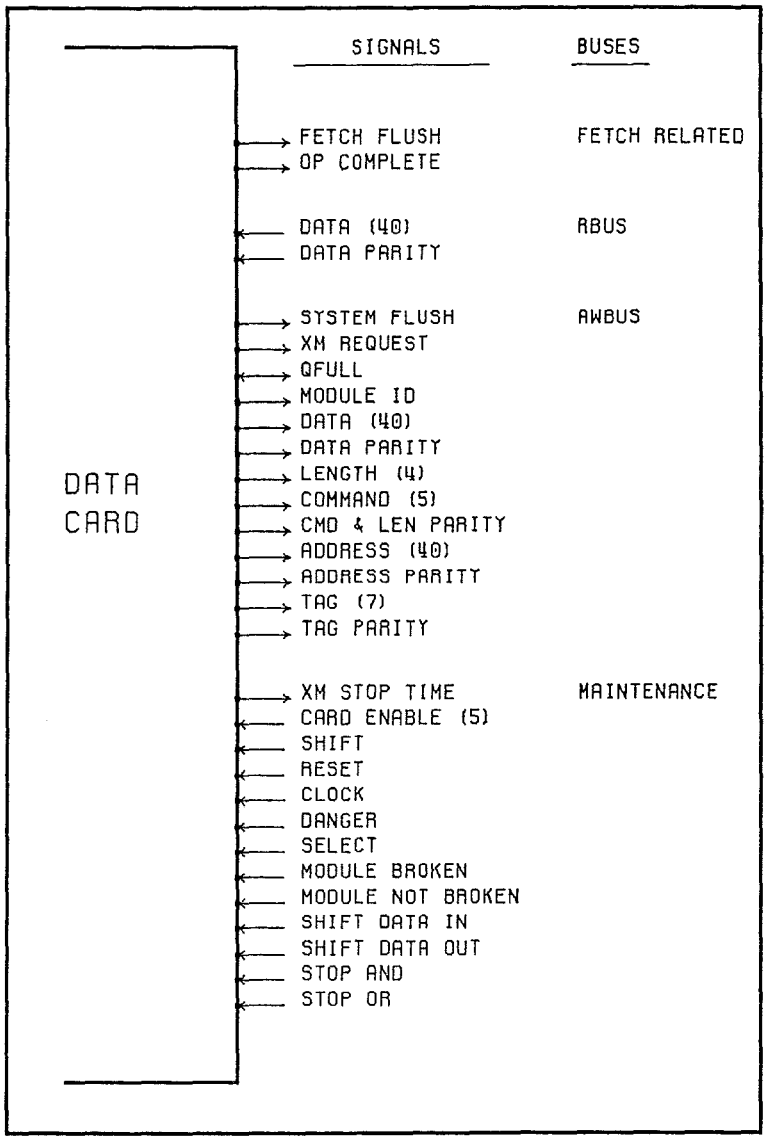


FIGURE 5-3 XMD BACKPLANE LOGIC SIGNALS



5.3 THE FETCH BUS (FBUS)

The FBUS connects the Fetch Module with the Execute Module. The Fetch module transmits 40 bits of data through Fbus1 and 10 bits of control information through Fbus2. Both queues reside on the control card.

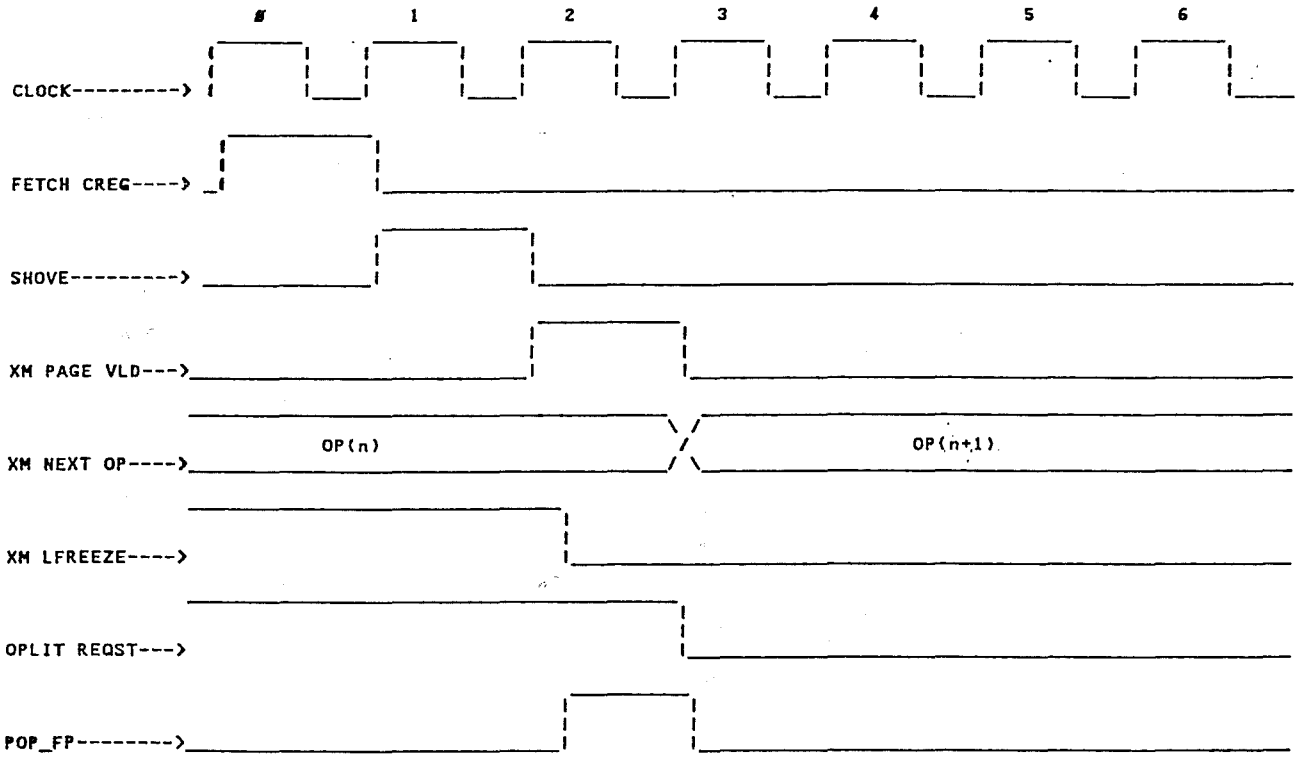
5.3.1 FBUS TIMING

The Fetch Module sends: its data, the Fetch queue address and appropriate load buffer signals, on the bus. The XM stores the information in its Fetch queue when clock is low. Exceptions are Shove, Optimal, and Data Hit which are registered at the rising edge of the next clock.

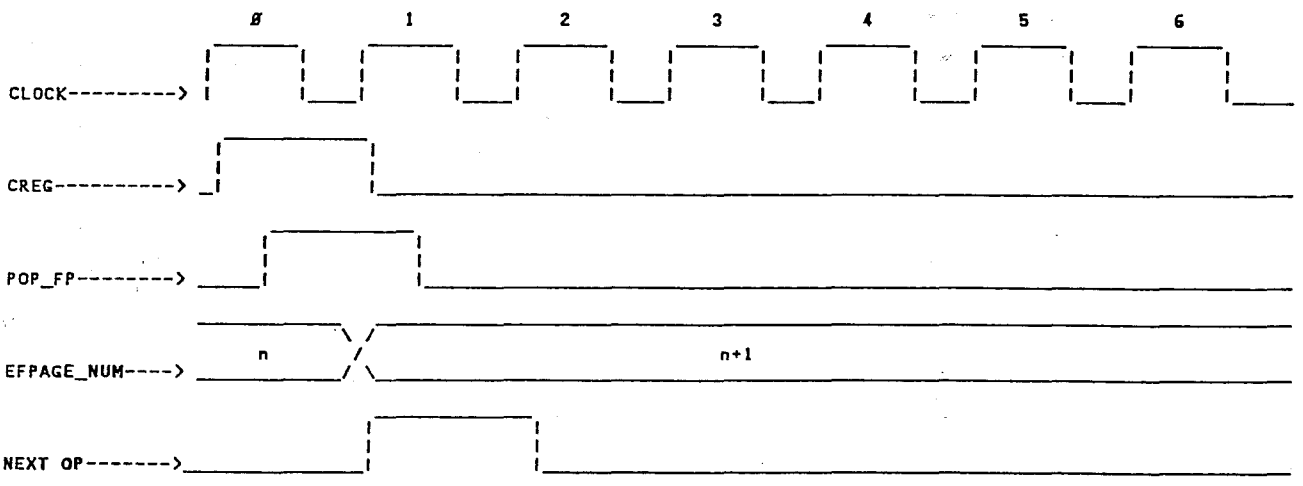
OPTIMAL and DATAHIT are registered only when SHOVE is high, otherwise they are ignored.

FIGURE 5-4 FBUS TIMING DIAGRAMS

FBUS TIMING DIAGRAM



OPLIT BRANCH TIMING DIAGRAM



5.3.2 FBUS LINE DESCRIPTION - 70 lines

The following paragraphs describe the use of the various lines comprising the Fbus interface.

FBUS1 [39:0]

This field transfers data with parity from the Fetch Module to the XM's Fetch queue 1. Information is primerily addresses or the Lit value in case of a Literal.

FBUS2 [9:0]

F2 Data is length and testable parameters relating to the instruction; also, Oplit.

FPADDR1 [1:0]

This two bit field specifies the page of the Fetch queue 1 into which Fetch data will be written.

FLOAD1

This signal enables the XM to load the data into Fetch queue 1.

FADDR1 [2:0]

The entry of the selected page of Fetch queue 1, which is to receive the data, is encoded in this 3 bit field.

5.3.2 FBUS LINE DESCRIPTION - 70 lines (Continued)

OFRQST

If the OF requests to drive the Fbuses, it drives the OFRQST high. The XM registers this signal solely for the purpose of isolating the driver in error if an error is detected on the bus.

FCPARITY1

Parity over FFADDR1, FADDR1, FLOAD1, and OFRQST is generated and passed along this wire (FCPARITY1).

FFADDR2 [1:0]

This two bit field specifies the page of the Fetch Queue 2 into which Fetch data will be written.

FLOAD2

This signal enables loading of data into Fetch Queue 2 on the XM control card.

FADDR2 [2:0]

The entry of the selected page of Fetch queue 2 which is to receive the data.

SHOVE

This signal specifies the end of the data transfer from Fetch for a given OP. It also sets "Page Valid"

5.3.2 FBUS LINE DESCRIPTION - 70 lines (Continued)

OPTIMAL

This line indicates that the instruction to be executed has satisfied certain constraints which allow it to be processed using optimised microcode. It may also use the Fetch requested operands which will be found in the XM's Operand Queue (OPQ). The criteria for deciding whether an instruction is optimal is a decision for XM firmware in conjunction with the Fetch designers. For details, see XM firmware EDS or Fetch module EDS.

DATA HIT

This line indicates that an interlock had existed on data requested by Fetch to be received by the OPQ. This means that the data may be in error and should be reread from memory. A conditional read, finding this signal true, will issue a read; finding it false, causes a no-op. Conditional read is described in Section 3.10.5.

FCPARITY2

Parity over FPADDR2, FADDR2, FLOAD2, SHOVE, OPTIMAL, and DATA HIT is generated and passed along this wire (FCPARITY2).

5.3.3 FBUS HARDWARE PROTOCOL

The XM hardware expects to find the F address in page entry 2 of Fetch Queue 1. The following are the Fetch Page formats in XM:

Entry #	FPQ1 Contents	FPQ2 Contents
0	OPSYL	OPLIT
1	ASYL	ALEN
2	BSYL	BLEN
3	CSYL	CLEN
4	PC	FLAGS
5	LENGTH SYL	-----
6	SPECIAL SYL	-----
7	TBS	-----

FPQ2 ENTRIES

OPLIT	BIT 9	=	Fetch page marked
	BIT 8	=	TBS
	BIT 7_1	=	Op Vector
	BIT 0	=	Literal Flag
ALEN	BIT 9_0	=	Length of A operand in digits
BLEN	BIT 9_0	=	Length of B operand in digits
CLEN	BIT 9_0	=	Length of C operand in digits
FLAGS	BIT 9_8	=	# of extended addresses
	BIT 7	=	AF > BF
	BIT 6	=	AF = BF
	BIT 5_4	=	A Address Controller (Ac)
	BIT 3_2	=	B Address Controller (Bc)
	BIT 1_0	=	C Address Controller (Cc)

5.4 AWBUS

The AWBUS provides the Execute Module and the Fetch Module the vehicle to send a read/write address, write data or status to the memory control.

5.4.1 AWBUS POLLING

The following is a description of the AWBUS Arbitration scheme.

BUS PRIORITY

The module with the highest priority is MCACM followed by XM, COP, OF, and IF which has the lowest priority in accessing the AWBUS.

MODULE ID

The following table lists the Module ID's for all the processor modules in the V500:

000	No transfer
001	XM
010	XM Co-processor
011	TBS
100	OF
101	IF
110	TBS
111	TBS

5.4.1 AWBUS POLLING (Continued)

BUS ARBITRATION SCHEME

Since XM has the highest priority on the AWBUS, it sends a request everytime it needs to. Along with the request, it drives the Module ID lines with the code 1. The only signal that can inhibit an XM request is QFULL sourced by MCACM. Refer to the MCACM specification, Section 5.1.3, for a more detailed and universal look at AWBUS access arbitration and error detection.

Only MCACM can inhibit Bus access by pulling on the QFULL line. The request signal from any module will not be inhibited.

Live freeze will not prevent the outstanding transfer in the one deep Memory Requestor queue from completing. Once the outstanding transfer is completed, the bus request F/F will reset itself and live freeze will prevent the F/F from setting again until the freeze is over.

5.4.2 AWBUS LINE DESCRIPTION - 104 lines

The following paragraphs describe the use of the various lines comprising the AWBUS interface.

XM REQUEST

This line is high when the XM requests the use of the bus. It has the second highest priority in accessing the bus. This signal inhibits the Fetch module from sending information to the bus.

MODULE ID [2:0]

The bus granted address is an error protection feature of the AWBUS arbitration network. The three lines give the address of the module that was granted access and driving the bus.

AWDATA [39:0]

This 10 digit data field transfers write data.

AWDPRTY

The write data parity bit shall be calculated such that the overall parity of the data field and the data parity bit is even.

AW LEN [3:0]

This four bit field in conjunction with the command field specifies the number of digits to be read/written from/to the memory control. The value of "0" indicates 10 digits. The values "A-F" are illegal.

5.4.2 AWBUS LINE DESCRIPTION - 104 lines (Continued)

AWBUS COMMAND [4:0]

This five bit field informs MCACM what action to take. See Section 5.4.6 for a list of the commands.

CONTROL PARITY

The control field parity shall be calculated such that the overall parity of the lergth, command and control parity bit is even.

AWADR [35:0]

This nine digit address field transfers read/write addresses to the MCACM. All addresses are relative to the particular base that is specified in the base indicant field.

AWBI [3:0]

This is the pointer to the entry in the Base/Limit table in the MCACM. The base indicant is the most significant digit of the ten-digit address field.

ADDRESS PARITY

The address field parity shall be calculated such that the overall parity of the base indicant, address field and the address parity bit is even.

5.4.2 AWBUS LINE DESCRIPTION - 104 lines (Continued)

TAG FIELD [6:0]

The tag field is an address which indicates to whom data is to be returned. Included is a receiver unit number as well as an entry number.

The following table describes the tags which the XM will recognize:

Mod Id #	Extension	Notes
1 1 0	0 n n n	XM RBUS Queue, nnn is the queue entry number
1 1 0	1 x x y	XM Operand Queue, xx is page number, y is the entry; CPOA is y=0, CPOB is y=1
1 0 1	0 i i x	XM write, ii is the current page number

This field has its own associated parity bit, giving even parity across the seven bits of tag and parity together.

QFULL

MCACM AW Queue full signal sent by MCACM when there is 1 more entry available in it's queue. XM registers this signal before using it in its bus request logic.

TAG PRTY

TAG parity calculated such that overall parity of the 7 bit tag field and tag parity is even.

5.4.3 READ COMMANDS

In case of a read operation, the XM sends out, on the AWBUS, the request together with the following information:

1. a read address through the address lines
2. a digit length field specifying the number of digits to be read (no more than 10 digits)
3. a command field specifying a read operation. parity of the data field and the data parity bit is even.
4. a tag field telling memory where to send the memory data.

The memory control will then send back the desired operand through the RBUS to the appropriate destination.

5.4.4 WRITE COMMANDS

In case of a write command, the module has to send out a request to the memory control together with the write address, the data to be written, the length field, and the command field specifying that the request is a write.

5.4.5 TAG GENERATION

Tags are used to indicate the intended destination of data returning from memory. Write normally doesnt expect return data, but in the case of an error will return an error descriptor. Section 5.4.2 contains a table showing the format of the tags.

RBQ ENTRY NUMBER

The RBQ entry number points to the one of eight entries in the RBUS queue to receive this particular piece of data. It is sent with each read request made by the XM except the conditional read.

OPQ ENTRY NUMBER

The OPQ entry number points to the particular entry of the Operand queue intended to receive the requested data. It is used by the XM only in the case of a conditional read. The three bits are assembled by looking at the two bits of the present Fetch page number and one bit encoded on whether the read is conditional read A or conditional read B.

5.4.6 AWBUS COMMANDS

Following is a list of the AWBUS commands:

AWBUS COMMAND	AWBUS COMMAND DESCRIPTION
01	Read Data
10	Write Data
16	Read with Lock
19	Write Base Table
1B	Write Limit Table
05	Read Base Table
07	Read Limit Table
13	Read Error Report
12	Read Error Address
10	Write FC
11	Read FC
02	Write I/O Register
03	Read I/O Register
18	Write-Back
06	Read ECC codes
1A	Write ECC codes
1E	Read Data Uncorrected
04	Test Error Detection
0C	Branch TAKEN
0D	Branch NTAKEN
0E	IF Flush (Generated only by 0F)
0F	System FLUSH

5.5 READ BUS (R-BUS)

The RBUS sends data from the memory control to the execute module and the Fetch module. It is single-sourced from the memory control.

The memory control sends out the requested data through the ten digit data lines, along with the data valid flag, one clock after the tag bits which act as a module load signal.

Back-to-back transfer can take place, but the firmware of the module has to restrict itself to have only eight outstanding transfers at any given time; otherwise, it will cause the eight deep RBUS queue to overflow which causes a "deadfreeze" to the module.

5.5.1 R-BUS LINE DESCRIPTION - 56 lines

The following paragraphs describe the RBUS lines relating to the XM.

RDATA [39:0]

This ten digit data field, with the associated parity bit, transfers the read data, memory error syndrome, etc., from the memory control to the requesting module.

RDPRTY

The read data parity bit shall be calculated such that the overall parity of the data field and the data parity bit is even.

RVALID

A bit sent with the data field to indicate, when high, that the data to be written into the queue is valid.

5.5.1 R-BUS LINE DESCRIPTION - 56 lines (Continued)

MCERROR [4:0]

The error information field contains error status indications of each operation initiated by the memory subsystem. A non-zero value in this field represents an error. Refer to the MCACM EDS for a detailed description of its encoding.

STATPRTY

The status parity bit shall be calculated such that the overall parity of the error information field, the data valid flag and the status parity bit is even.

RTAG [6:0]

Indicate the intended destination of the data which will be sent on the next clock,including the specific entry number of that Queue. XM can receive data into its 8 deep RBus Queue or into its 8 deep Operand Queue. Section 5.4.2 contains a table of the tags showing their formats.

RTPRTY

This is an exact copy of the parity bit sent by the processor to MCACM. It shall be calulated such that the overall parity of the tag field and the tag parity bit is even.

5.5.2 TAG HANDLING

Tags come back one clock ahead of the data they are associated with. Each is decoded into a write enable for the proper entry of the proper queue for the next clock. A separate register is used to receive error vectors.

5.6 NON-BUSSED LINE DESCRIPTION

The following signals are intermodule command and indicator lines.

5.6.1 SYSTEM FLUSH

Signal from XM to Fetch calling for a full system flush. An environment change is indicated by activating this line. (See Appendix F for details.)

5.6.2 FETCH FLUSH

Signal from XM to Fetch calling for a flush of Fetch only. It is active when a mispredicted branch is executed. (See Appendix F for details.)

5.6.3 XM_BRANCH_OK

If the XM encounters a branch instruction that is predicted TAKEN and it does take a branch, then it activates this wire. XM_BRANCH_OK is active high.

5.6.4 XM_OP_COMPLETE

This signal from XM to FETCH indicates the completion of a write stream. It does not necessarily incicate completion of execution. You can have more thar one op-complete per instruction.

5.6.5 XM_LIX_OK

This signal informs Fetch that an LIX instruction has just written the index registers.

5.6.6 XM_TIMEOUT

This signal is set 1.3 seconds after an OPLIT branch. Fetch and XM receive this wire to branch on.

5.6.7 FOP FETCH PAGE

This line is activated by the XM when it has finished an instruction, to indicate to Fetch that another page is available for a new instruction. After a maintenance RESET or a FLUSH, the microcode will get started by performing an OPLIT branch. At this time, the hardware will also activate FPOPFQUE$P to get to the next available FETCH page.

5.6.8 XM EVENT

A dedicated wire between Fetch and XM, set by XM via an internal command. For microcode use only. (Refer to XM microcode EDS.)

5.6.9 FETCH EVENT

A dedicated wire between Fetch and XM, set by OF. For microcode use only. (Refer to XM microcode EDS.)

5.7 INTERRUPTS: FAULTS, ERRORS

5.7.1 CONDITIONS THAT INTERRUPT XM:

	SOURCE	MASKABLE*
Normal IO Complete	IOMC	Yes
IO Error Complete	IOMC	"
Real Time IO Complete	IOMC	"
Task Timer Interrupt	Timer	"
Task Timer Fault	Timer	"
SMC Interrupt	Stop	NO
Fault	FRGCTL	"
Trace/Debug Interrupt	mode	"
IPC	ext	"
Snap Picture Taken	chain	"

* Mask bit is set to 1 to enable the interrupt condition.

5.7.2 HANDLING OF INTERRUPTS, FAULTS, AND ERRORS

Fetch detected faults will be handled by converting the op to a 1E code and passing it to XM. When XM executes the instruction it will do a Hardware Call Procedure (HCP).

XM interrupts will be handled at the end of the current instruction if the interrupt is enabled.

Most XM errors will be handled immediately by deadfreezing and allowing the SMC to set up an instruction retry. CRAM errors will cause a live freeze and wait for the SMC to do a micro-instruction retry.

MCACM error information will be decoded by XM regardless of the destination of the data or whether it is for the current instruction or a future one. The decode must determine the destination, whether it is for the current or a future instruction, and whether it is a fault, an error or a corrected error.

5.7.2 HANDLING OF INTERRUPTS, FAULTS, AND ERRORS (Continued)

Of the errors reported in the MCACM error information field, only Undigits in Address and Limit Error are failures that will result in an HCP.

Faults returning to Fetch will use the 1E/1A op path.

Faults returning to the XM will be handled when they are popped from the queue. When data with a fault is popped, the XM fault interrupt F/F is set. The F/F can be tested by the microcode or will cause a branch to the interrupt code when an OPLIT branch is attempted. The interrupt microcode will determine whether the fault was detected by XM microcode or the MCACM.

If the MCACM error information field indicates an error for a future instruction and the Notry flipflop is reset, it will be handled immediately by deadfreezing and calling the SMC to do an instruction retry. If the error is for a future instruction and the notry flipflop is set, it will be handled by retry when the current instruction is finished. If another error occurs, it will replace the earlier one on a first-worst basis. If the error is for the current instruction, it is handled immediately by deadfreezing. A MCACM to XM interface parity error is treated the same as if it came in the error information field. In each of the above cases, it is up to the SMC to determine what type of recovery should be attempted while the XM hardware determines when and what type of stop to do. (Refer to Appendix B for details.)

Fetch-XM interface errors set FETCHERR.

5.8 IOMC INTERFACE

5.8.1 IO COMPLETE SIGNALS

The IO Complete signals are sourced by the IOMC and bussed to all of the XM's.

REAL TIME IO COMPLETE

Indicates that a Real Time IO operation has completed.

NORMAL IO COMPLETE

Indicates that a Normal IO operation has completed.

ERROR IO COMPLETE

Indicates that an IO has terminated with an error.

5.8.2 XM POLL

Sourced by the IOMC. Indicates that the IO Complete lines are valid. True for two clocks.

5.8.3 PARITY

Even parity accross the IO Complete lines. Generated by the IOMC. Parity is checked only if the particular XM is being polled, otherwise no checking is done.

5.8.4 TAKEN and NOT TAKEN

These lines are used to respond to the poll, they are bussed to all the XM's and IOMC's.

5.8.5 XM PRESENT

Indicates to the IOMC that the TAKEN and NOT TAKEN are valid.

5.8.6 IOMC INTERFACE OPERATION

Each XM has three mask bits in the interrupt mask register that indicate the type of IO complete it is armed to service. There are three more bits that indicate the type of IO complete interrupts the interface has accepted. All six of the flags can be set and cleared by the microcode. Each XM has a dedicated poll line. An IOMC polls the XM's one at a time via the four poll lines (one for each XM) along with polling the XM's. IOMC will indicate which type of IO complete has occurred. The selected XM will respond to the poll by asserting XM PRESENT and either one of TAKEN or NOT TAKEN (but not both). If the interface is enabled to receive any of the IO completes, it will accept all that are indicated, and will interrupt the module at Oplit branch.

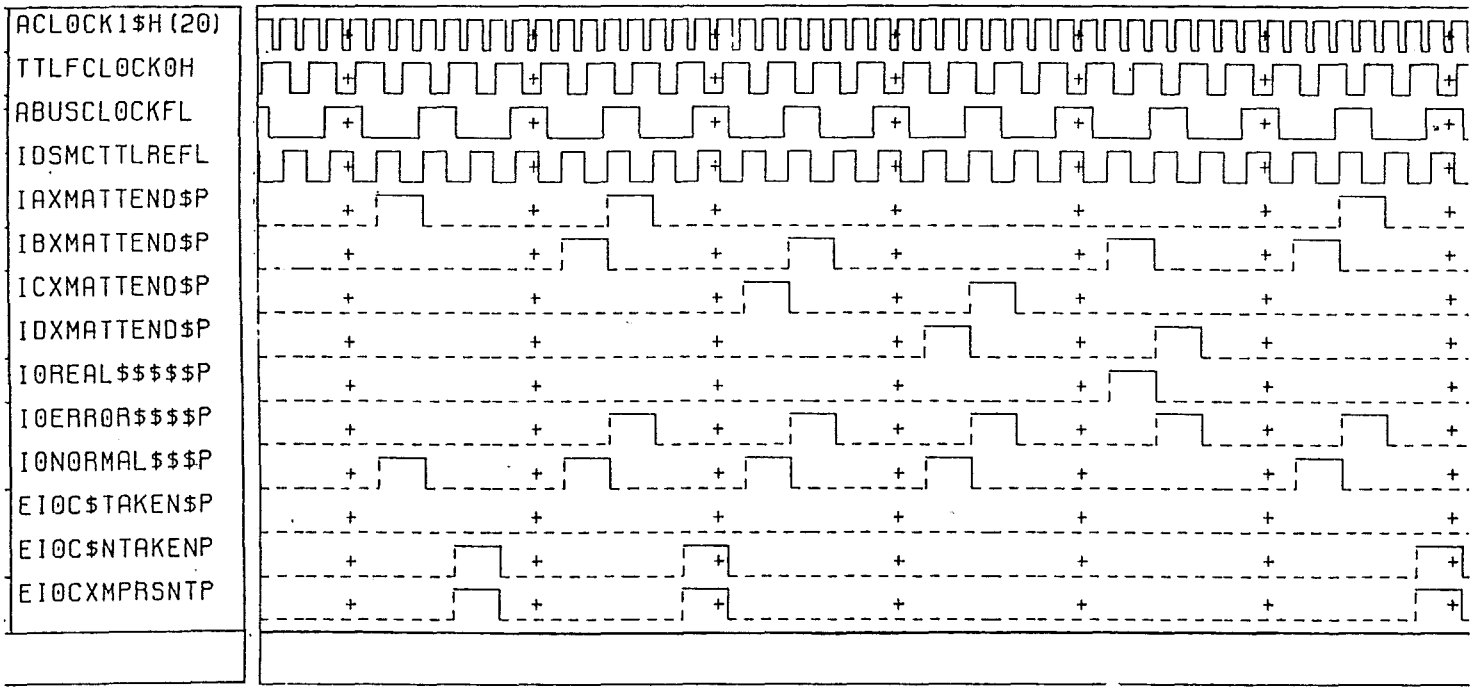
The following table defines all possible combinations driven out by XM in response to a poll.

XM PRESENT	IO TAKEN	IO NTAKEN	MEANING
0	0	0	OFFLINE/XM DEAD/ XM NOT PRESENT
0	0	1	Incorrect Operation
0	1	0	Incorrect Operation
0	1	1	Incorrect Operation
1	0	0	Incorrect Operation
1	0	1	NOT TAKEN IO
1	1	0	TAKEN IO
1	1	1	Incorrect Operation

The entire interface is based on two clocks per any transaction. Refer to the timing diagram and explanation in Figure 5-5. The two clock timing is administered by a signal from SMC called AMFZBUSREFOP which is registered inside XM before being used.

The following conditions will disable the IOMC interface.

A_ XM DEAD (internal error condition)
B_ XM OFFLINE (via the maintenance chair set by SMC)
C_ XM OK2STOP (CREG parity error, Processor HALT, FUTURE error, STOP event)



TRANSCRIPT OF I/O COMPLETE INTERFACE TIMING EXAMP

1 SET XM-A ONLINE
SET XM-B OFFLINE
SET XM-C OFFLINE
SET XM-D OFFLINE
START A NORMAL IOC ON IOMC-A
IOMC-A POLLS XM-A WITH NORMAL IOC: NOT TAKEN

2 START AN ERROR IOC ON IOMC-B
IOMC-A POLLS XM-B WITH NORMAL IOC: OFFLINE
IOMC-B POLLS XM-A WITH ERROR IOC: NOT TAKEN

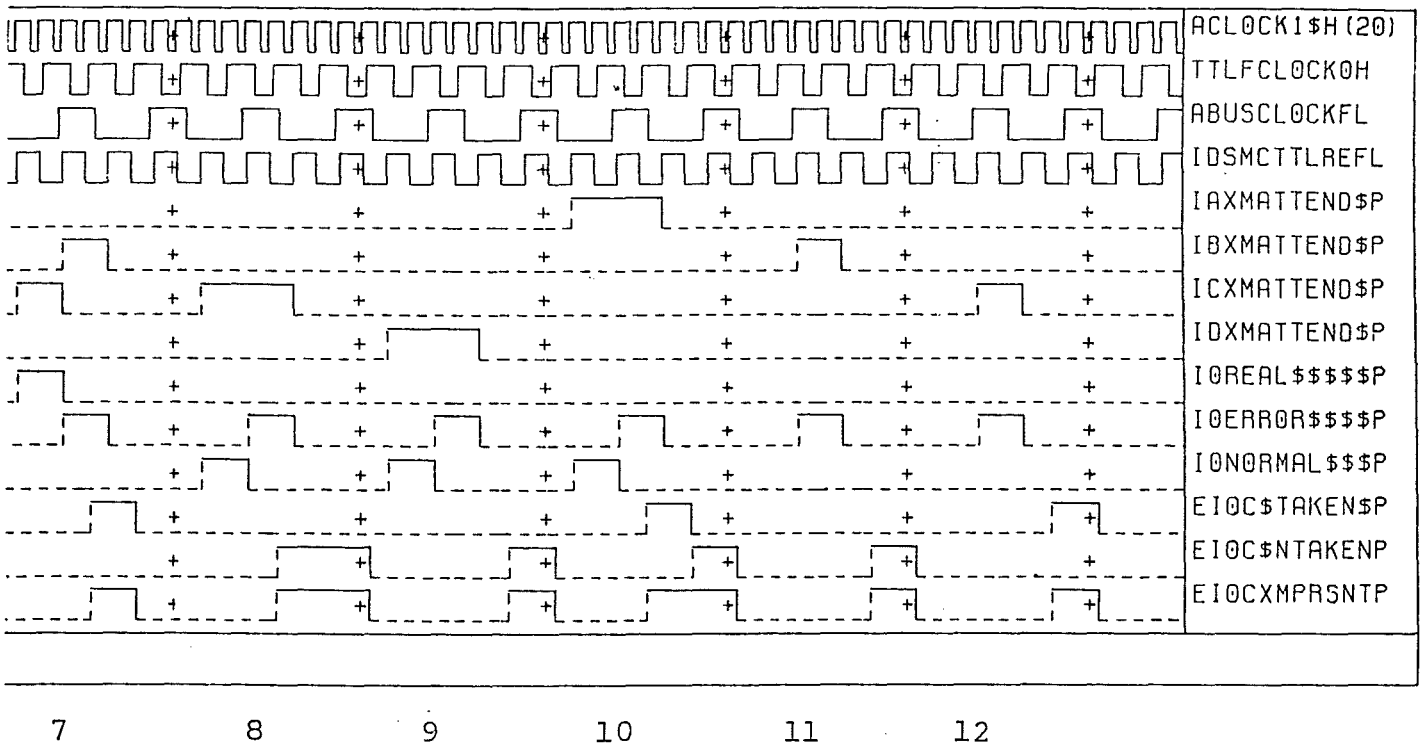
3 IOMC-A POLLS XM-C WITH NORMAL IOC: OFFLINE
IOMC-B POLLS XM-B WITH ERROR IOC: OFFLINE

4 IOMC-A POLLS XM-D WITH NORMAL IOC: OFFLINE
IOMC-B POLLS XM-C WITH ERROR IOC: OFFLINE

5 SET XM-C ONLINE
START A REAL IOC ON IOMC-A
IOMC-A POLLS XM-B WITH REAL IOC: OFFLINE
IOMC-B POLLS XM-D WITH ERROR IOC: OFFLINE

6 ENABLE XM-C FOR REAL IOC
IOMC-A POLLS XM-B WITH NORMAL IOC: OFFLINE
IOMC-B POLLS XM-A WITH ERROR IOC: NOT TAKEN

FIGURE 5-5 IOC TIMING EXAMPLE



7 IOMC-A POLLS XM-C WITH REAL IOC: TAKEN
IOMC-B POLLS XM-B WITH ERROR IOC: OFFLINE

8 IOMC-A POLLS XM-C WITH NORMAL IOC: NOT TAKEN
IOMC-B POLLS XM-C WITH ERROR IOC: NOT TAKEN

9 IOMC-A POLLS XM-D WITH NORMAL IOC: OFFLINE
SET XM-D ONLINE
SET XM-B ONLINE
IOMC-B POLLS XM-D WITH ERROR IOC: NOT TAKEN

10 ENABLE XM-D FOR ERROR IOC
ENABLE XM-A FOR NORMAL IOC
IOMC-A POLLS XM-A WITH NORMAL IOC: TAKEN
IOMC-B POLLS XM-A WITH ERROR IOC: NOT TAKEN

11 IOMC-B POLLS XM-B WITH ERROR IOC: NOT TAKEN

12 ENABLE XM-C FOR ERROR IOC
IOMC-B POLLS XM-C WITH ERROR IOC: TAKEN

FIGURE 5-5 IOC TIMING EXAMPLE (Continued)

6 CLOCK CIRCUITRY

The XM runs on a 48ns clock with a 10% margin. The clock is gated to retain data integrity after errors, to aid in diagnostics (See 7.3), for breakpoint generation (See Section 7.2), and to produce synchronizing wait states without the use of uCode wait loops.

6.1 SELF STOP CONDITIONS

In the event the XM wishes to stop itself, it uses the Hold function of the registers to prevent any change of state to those pieces of logic that respond to one of the two conditions listed below.

1) Livefreeze: a continuable stop; the XM is partially frozen, but all data and state is preserved
2) Deadfreeze: a catastrophic stop of the entire XM, wherein data or state or both are lost

6.1.1 LIVEFREEZE

Livefreeze takes place when a continuation is required after the condition causing the stop goes away. This assumes that the stop condition can go away, that the module data integrity has been maintained, and that no bad transmissions of data took place as a result of the stop.

Livefreeze is used under the following circumstances:

1) uPC address stop (See 7.2.)
2) Read RBUS or OPQ queue when queue entry is not valid
3) Request AWBUS and previous request denied
4) OpLit branch when Fetch page not valid
5) CREG parity error

In Case 1 and 5, the maintenance processor must disable the stop and also reset the condition which caused it. In Case 5, it must also rewrite the bad microword. Case 2 is removed when awaited data is received from the RBUS. Case 3 is removed when access is eventually granted. Case 4 is removed when Fetch sets Fetch page valid.

To prevent data from being lost while a module is in livefreeze several safeguards have been established.

The safeguards are:

1) The RBUS queue and OPQ are not live-frozen

2) The loading of the Memory Requestor interface is frozen but the transmission and resetting of the bus request is not live-frozen

3) The Fetch queue is not live-frozen

4) The firmware programmers have been advised to limit the number of outstanding requests for return data on the RBUS to 8 (the number of entries in the RBUS queue). Failure to obey this rule results in the RBUS queue overflow condition, which generates deadfreeze.

6.1.2 DEADFREEZE

Deadfreeze is used to stop the module after a fatal error is detected. The clock is stopped one clock after the detection of the error because errors can not be resolved until after the nondecision edge of the clock. The module will not be able to recover without intervention in the form of a pipeclear or maintenance reinitialization. The deadfreeze bit is set by the condition causing the error.

See 7.3 "Self Stop" for details of XM error detection.

6.2 STORAGE ELEMENT CLOCKING

Two types of storage elements are used in the XM:

1) Positive edge triggered registers

2) Level sensitive latch structures

There are several forms of the type 2 storage elements, the mass store, the mem requestor address file, the input queues, the litfile, the programmable Ptest functions, the CRAM, and the stack. These are described as hidden state because they are not directly viewable on shift chains.

6.2.1 REGISTER CLOCKING

The registers are clocked with the standard XM clock.

6.2.2 CRAM WRITE

There are three pieces of information that need to be shifted in the chain to perform a Control Store Write. The data to be written is shifted into the least sionificant 134 bits (CREG). The address to be written is shifted into the next 14 bits in the chain on the CSADR register. Finally, the write enable bit, CRAMWE, is shifted in with a "1". To complete the write cycle, the module waits for the RAMWRTEN pulse from the SMC.

6.2.3 STACK CLOCKING

The Stack requires a write enable pulse which is satisfied by the standard XM clock.

6.3 RAM STATE INITIALIZATION

Ram in the XM falls into two catecories by the way it is used; Read/Write and Read only. Read only RAMs need to be loaded by maintenance prior to running any ucode. Read/write ram may be initialized for fault detection purposes.

6.3.1 LITFILE INITIALIZATION

Litfile will be loaded by shift chain setting of address in the CREG litfile address field, putting data to be loaded into the Dreg, setting Litfile Write Enable (LFWE) on the chain, and the RAMWRTEN pulse. LFWE will reset itself after the next clock.

6.3.2 PTEST INITIALIZATION

PTEST ram data will be placed in the DREG, address of one byte replicated 5 times in the AUREG, function number in the PTEST function select field of the CREG (CREG[28:5]), the PTWE bit on the shift chain set, and the RAMWRTEN pulse. The PTWE (PTEST Write Enable) flip-flop is wired to reset on any normal clock.

6.3.3 FETCH QUEUE INITIALIZATION

The Fetch Queue should have been written into by Fetch before the XM can read it. A read attempt, before Fetch sets the page valid, will freeze the XM; therefore, no initialization will be done. Maintenance may wish to write data with proper parity to prevent parity error reports during maintenance read just after power-on.

6.3.4 SCRATCHPAD INITIALIZATION

The XM should have written into a scratchpad location before it attempts to read it; therefore, there is no hardware initialization capability provided for the scratchpad. Maintenance may want to write good parity into the scratchpad to prevent parity errors from being reported during maintenance reads.

6.3.5 MASS STORE INITIALIZATION

Like scratchpad, the Mass Store will have no hardware initialization but maintenance may want to initialize at power up using shift chain load routines.

6.3.6 RBUS QUEUE INITIALIZATION

The Rbus queue may not be read by the XM until valid data is returned from memory; therefore, initialization is not necessary. Any attempt to read RBQ before valid data arrives, will cause a live freeze.

6.3.7 OPERAND QUEUE INITIALIZATION

The Operand Queue, like the Rbus Queue, causes a freeze if read before valid data is returned to it. It does not require initialization.

7 MAINTENANCE FEATURES

Maintenance considerations in the XM include the following:

1) All registers on shift chains
2) Stop logic
3) Detection of XM internal errors
4) Branch history file
5) Data register on the scratchpad input
6) Parity on external Busses
7) All hidden state storage accessible directly from registers except the requestor length scratchpads.

7.1 SHIFT CHAINS

There are 4 shift chains per XM. The control board has the control chain along with the maintenance chain. Resident on the data card is the data chain and its maintenance chain. The maintenance chain contents can be altered dynamically; whereas, the other two chains require that the module be stopped. Below is a list of the actual location of the various fields within each chain and their corresponding length.

```
                       XM CONTROL CARD CHAIN LIST
                       --------------------------

    CHAIN   EC;                           %615 BITS LONG.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %       TIMER.  EC[614-565]                                     %
    %       ------------------                                      %
    %       CHAIN LENGTH --> 50 BITS.                               %
    %       LOCATION ------> D2N0.                                  %
    %       SDI PIN -------> R06.                                   %
    %       SDO PIN -------> R02.       PART # : 1995 6507          %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        TIMERERR        1;                % TIMER DETECTED AN ERROR.
        TOUTCTR         3;                % TIMEOUT COUNTER.
        TOUTPRTY        1;                % TIMEOUT COUNTER PARITY.
        TASKPRTY        1;                % TASK TIMER PARITY.
        TASKTIMR       32;                % TASK TIMER COUNTER.
        ERROKCOPY       1;                % ERROR OK FROM FORCE ERROR.
        FORCER          1;                % FORCE ERROR.
        USECPRTY        1;                % MICROSECOND COUNTER PARITY.
        USECCTR         4;                % MICROSECOND COUNTER.
        USECSYNC        2;                % MICROSECOND SYNC BIT.
        USECSYNCB       2;                % DUPLICATE USECOND SYNC BIT.
        CRAMWE          1;                % CONTROL STORE WRITE ENABLE.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %       FRZCTL4.  EC[564-513]                                   %
    %       --------------------                                    %
    %       CHAIN LENGTH --> 52 BITS.                               %
    %       LOCATION ------> 1DF8.                                  %
    %       SDI PIN -------> H15.                                   %
    %       SDO PIN -------> L01.       PART # : 1996 2489          %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        UNDIGAU         1;                % UNDIGIT AT THE AU INPUTS.
        NONFATAL        1;                % MCACM NFATAL ERROR INDICATOR.
        DFRZDIS         1;                % DEAD FREEZE DISABLE.
        STOPEVENT       1;                % STOP EVENT INDICATOR.
        REMEMBER        1;                % STOP EVENT REMEMBER FLAG.
        POPFPAGE        1;                % POP FETCH PAGE.
```

```
BPSTOPAND       1;      % REGISTERED BP STOP AND.
BPSTOPOR        1;      % REGISTERED BP STOP OR.
FETCHERR        1;      % FETCH ERR DETECTED INDICATOR.
DATA-HIT        1;      % DATA HIT FOR THE CURRENT PAGE.
CREADA          1;      % CONDITIONAL READ A INDICATOR.
CREADB          1;      % CONDITIONAL READ B INDICATOR.
FAKEDFRZ1       1;      % 2ND CYCLE OF FAKE DFRZ.
FAKEDFRZ0       1;      % 1ST CYCLE OF FAKE DFRZ.
FRZCODEPTY      1;      % FREEZE CODE PARITY.
FRZCODE         4;      % HOLDS THE FREEZE CODE.
                        % (USED FOR FUNCTION TEST)
TESTCODE        5;      % HOLDS THE TEST CODE.
                        % (USED FOR FUNCTION TEST)
NORDATA         1;      % NO VALID RBUS/OPQ ENTRY.
FUTURERR        1;      % FREEZE DETECTED SOME CLOCKS
                        % AGO.(FETCH RELATED)
NOFPAGE         1;      % NO VALID FETCH PAGE.
AWBUSY          1;      % AW BUS REQUEST DENIED.
CREGERR         1;      % CREG PARITY ERROR.
XMDEAD          1;      % XM DEAD FREEZE INDICATOR.
PTESTERROR      1;      % REGISTERED PTEST ERROR.
FPQ2ERROR       1;      % FPQ2 SLICE 0 & 1 ERROR.
FPBERROR        1;      % PARITY ERROR ON FPB PORT.
FPAERROR        1;      % PARITY ERROR ON FPA PORT.
ERROROK         1;      % ERROR IGNORE REGISTER.
PMUXBERROR      1;      % RE-REGISTERED PMUXB ERROR.
PMUXAERROR      1;      % RE-REGISTERED PMUXA ERROR.
RWAU0ERROR      1;      % RE-REGISTERED REWAU0 ERROR.
RWAU1ERROR      1;      % RE-REGISTERED REWAU1 ERROR.
REAM0ERROR      1;      % RE-REGISTERED REAM0 ERROR.
REAM1ERROR      1;      % RE-REGISTERED REAM1 ERROR.
RCTAGERROR      1;      % RE-REGISTERED RECTAG ERROR.
FRQERRDIS       1;      % FRQCTL ERROR DISABLE.
TIMERERROR      1;      % RE-REGISTERED TIMER ERROR.
SEQ2ERROR       1;      % RE-REGISTERED SEQ2 ERROR.
SEQ1ABERR       1;      % RE-REGISTERED SEQ1A/B ERROR.
SEQ1CDERR       1;      % RE-REGISTERED SEQ1C/D ERROR.
FRQCTERROR      1;      % RE-REGISTERED FRQCTL ERROR.
INTPBERROR      1;      % RE-REGISTERED INTRPTB ERROR.
INTPAERROR      1;      % RE-REGISTERED INTRPTA ERROR.
XMD1ERROR       1;      % REGISTERED XMD ERROR.
                        % (MOST SIGNIFICANT HALF)
XMD0ERROR       1;      % REGISTERED XMD ERROR.
                        % (LEAST SIGNIFICANT HALF)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        INTRPT2. EC[512-447]          (SLAVE COPY)              %
%        ------------------                                      %
%        CHAIN LENGTH --> 66.                                    %
%        LOCATION ------> IONO.                                  %
%        SDI PIN -------> P11.                                   %
%        SDO PIN -------> M15.         PART # : 1996 1606        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                                       % INTRPT ARRAY #1.(SLICE B)
    ERRORB          1;                 % FATAL HARDWARE ERROR
    CLR.SMCINT      1;                 % CLEAR THE SMCINT BIT IN STOP
                                       % ARRAY.
                                       %     internal commands
                                       %     ---------------
    SETXMEVNTB      1;                 % SET XM EVENT.
    SET.IPCB        1;                 % SET INTER-PROC COMM BIT.
    BRANCH.OKB      1;                 % BRANCH OK.
    LIX.OKB         1;                 % LOAD INDEX OK.
    IFBUSLDB        1;                 % INTERNAL FBUS LOAD ENABLE.
                                       %     -------------------
    LATE.LFRZB      1;                 % LIVE FREEZE DELAYED 1 CLOCK.
                                       %     coprocessor interface
                                       %     -------------------
    COPPARITYB      1;                 % COPROCESSOR DATA PARITY.
    COPDATAB-0      1;                 % COPROCESSOR DATA BIT 0.
    COPDATAB-1      1;                 %     "          "   "  1.
    COPDATAB-3      1;                 %     "          "   "  3.
    COPDATAB-2      1;                 %     "          "   "  2.
    COPVALIDB       1;                 % COPROCESSOR DATA VALID.
                                       %     IOMC interface
                                       %     ------------
    UNUSEDIO        7;                 % UNUSED BITS IN THIS SLICE.
                                       %     interrupt bit
                                       %     ------------
    INTERRUPTB      1;                 % INTERRUPT.
    INTINHIBTB      1;                 % INTERRUPT INHIBIT.
                                       %     interrupt register B
                                       %     ------------------
    SNAPICTAKB      1;                 % SNAP PICTURE TAKEN INTERRUPT.
    IPCB            1;                 % INTER-PROC COMM INTERRUPT.
    TRACDEBUGB      1;                 % TRACE OR DEBUG INTERRUPT.
    FAULTB          1;                 % FAULT INTERRUPT.
    SMC.INTB        1;                 % SMC INTERRUPT.
```

```
                                    %      timeout control
                                    %      ---------------
      TOGATEB          1;           % TIMEOUT ONE SHOT GATE.
                                    %      interrupt mask register
                                    %      -----------------------
      TIMOUTMSKB       1;           % TIMEOUT 1 MASK.
      TTF.MASKB        1;           % TASK TIMER FAULT MASK.
      TTI.MASKB        1;           % TASK TIMER INTERRUPT MASK.
      REAL.MASKB       1;           % REAL TIME IO COMPLETE MASK.
      ERR.MASKB        1;           % ERROR IO COMPLETE MASK.
      NORM.MASKB       1;           % NORMAL IO COMPLETE MASK.
                                    %      interrupt register A
                                    %      --------------------
      TIMEOUT.2B       1;           % TIMEOUT 2.
      TIMEOUT.1B       1;           % TIMEOUT 1.
      TSKTIMFLTB       1;           % TASK TIMER FAULT INTERRUPT.
      TSKTIMINTB       1;           % TASK TIMER INTERRUPT.
      REAL.IOCB        1;           % REAL TIME IO COMPLT INTERRUPT.
      ERROR.IOCB       1;           % ERROR IO COMPLETE INTERRUPT.
      NORM.IOCB        1;           % NORMAL IO COMPLETE INTERRUPT.
                                    %      test conditions
                                    %      ---------------
      FECHEVENTB       1;           % FETCH EVENT.
      MISC.FF1B        1;           % MISCELLANEOUS FLIP-FLOP 1.
      MISC.FF0B        1;           % MISCELLANEOUS FLIP-FLOP 0.
      GOVFLOWB         1;           % GLOBAL OVERFLOW.
      LOVFLOWB         1;           % LOCAL OVERFLOW.
      GCOMLB           1;           % GLOBAL COM LOW.
      GCOMHB           1;           % GLOBAL COM HIGH.
      LCOMLB           1;           % LOCAL COM LOW.
      LCOMHB           1;           % LOCAL COM HIGH
                                    %      miscellaneous register
                                    %      ----------------------
      SYSFLUSHB        1;           % SYSTEM FLUSH.
      FETFLUSHB        1;           % FETCH FLUSH.
      NORTYB           1;           % NO RETRY.
      SOFTSNAPB        1;           % SOFT SNAP.
      PRO.HALTB        1;           % PROCESSOR HALT.
      MOPOKB           1;           % MEASUREMENT REGISTER VALID.
      LITERAL2B        1;           % LIT BIT FOR NON_OP_FP BRANCHES
                                    %      mode register
                                    %      -------------
      PRO.STAB         3;           % PROCESSOR STATE.
      DEBUG.ENB        1;           % DEBUG INTERRUPT ENABLE.
      SOFTFAULTB       1;           % SOFT FAULT.
      PRIVILGEB        1;           % PRIVILEGED MODE.
      TRACE.ENB        1;           % TRACE INTERRUPT ENABLE.
      SNAP.ENB         1;           % SNAP ENABLE.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        INTRPT2. EC[446-381]          (MASTER COPY)                         %
%        ------------------                                                  %
%        CHAIN LENGTH --> 66 BITS.                                           %
%        LOCATION ------> IOK6.                                              %
%        SDI PIN -------> P11.                                               %
%        SDO PIN -------> M15.         PART # : 1996 1606                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                                       % INTRPT ARRAY # 2.(SLICE A)
     ERRORA          1;                % FATAL HARDWARE ERROR.
     USECPULSE       1;                % 1 USEC PULSE FROM IOMC.
                                       %     internal commands
                                       %     -----------------
     SETXMEVNTA      1;                % SET XM EVENT.
     SET.IPCA        1;                % SET INTER-PROC COMM BIT.
     BRANCH.OKA      1;                % BRANCH OK.
     LIX.OKA         1;                % LOAD INDEX OK.
     IFBUSLDA        1;                % INTERNAL FBUS LOAD.
                                       %     -------------------
     LATE.LFRZA      1;                % LIVE FREEZE DELAYED 1 CLOCK.
                                       %     coprocessor interface
                                       %     ---------------------
     COPPARITYA      1;                % COPROCESSOR DATA PARITY.
     COPDATAA        4;                % COPROCESSOR DATA.
     COPVALIDA       1;                % COPROCESSOR DATA VALID.
                                       %     IOMC interface (not used)
                                       %     -------------------------
     TAKENA          1;                % IO COMPLETE TAKEN.
     IOMC.ERROR      1;                % PARITY ERROR ON IOMC LINES.
     REALIOCRA       1;                % REAL TIME IO COMPLETE.
     ERR.IOCRA       1;                % ERROR IO COMPLETE.
     NORMIOCRA       1;                % NORMAL IO COMPLETE.
     IOMC.POLL       1;                % POLL LINE FROM IOMC.(ATTEND)
     BUSREF          1;                % REGISTERED BUS REFERENCE.
                                       %     interrupt bit
                                       %     -------------
     INTERRUPTA      1;                % INTERRUPT.
     INTINHIBTA      1;                % INTERRUPT INHIBIT.
                                       %     interrupt register B
                                       %     ----------------------
     SNAPICTAKA      1;                % SNAP PICTURE TAKEN INTERRUPT.
     IPCA            1;                % INTER-PROC COMM INTERRUPT.
     TRACDEBUGA      1;                % TRACE OR DEBUG INTERRUPT.
     FAULTA          1;                % FAULT INTERRUPT.
     SMC.INTA        1;                % SMC INTERRUPT.
```

```
                                    %     timeout control
                                    %     ---------------
    TOGATEA         1;              % TIMEOUT ONE SHOT GATE.
                                    %     interrupt mask register
                                    %     -----------------------
    TIMOUTMSKA      1;              % TIMEOUT 1 MASK.
    TTF.MASKA       1;              % TASK TIMER FAULT MASK.
    TTI.MASKA       1;              % TASK TIMER INTERRUPT MASK.
    REAL.MASKA      1;              % REAL TIME IO COMPLETE MASK.
    ERR.MASKA       1;              % ERROR IO COMPLETE MASK.
    NORM.MASKA      1;              % NORMAL IO COMPLETE MASK.
                                    %     interrupt register
                                    %     ------------------
    TIMEOUT.2A      1;              % TIMEOUT 2.
    TIMEOUT.1A      1;              % TIMEOUT 1.
    TSKTIMFLTA      1;              % TASK TIMER FAULT INTERRUPT.
    TSKTIMINTA      1;              % TASK TIMER INTERRUPT.
    REAL.IOCA       1;              % REAL TIME IO COMPLT INTERRUPT.
    ERROR.IOCA      1;              % ERROR IO COMPLETE INTERRUPT.
    NORM.IOCA       1;              % NORMAL IO COMPLETE INTERRUPT.
                                    %     test conditions
                                    %     ---------------
    FECHEVENTA      1;              % FETCH EVENT.
    MISC.FF1A       1;              % MISCELLANEOUS FLIP-FLOP 1.
    MISC.FF0A       1;              % MISCELLANEOUS FLIP-FLOP 0.
    GOVFLOWA        1;              % GLOBAL OVERFLOW.
    LOVFLOWA        1;              % LOCAL OVERFLOW.
    GCOMLA          1;              % GLOBAL COM LOW.
    GCOMHA          1;              % GLOBAL COM HIGH.
    LCOMLA          1;              % LOCAL COM LOW.
    LCOMHA          1;              % LOCAL COM HIGH.
                                    %     miscellaneous register
                                    %     ----------------------
    SYSFLUSHA       1;              % SYSTEM FLUSH.
    FETFLUSHA       1;              % FETCH FLUSH.
    NOTRYA          1;              % NO RETRY.
    SOFTSNAPA       1;              % SOFT SNAP.
    PRO.HALTA       1;              % PROCESSOR HALT.
    MOPOKA          1;              % MEASUREMENT REGISTER VALID.
    LITERAL2A       1;              % LIT BIT FOR NON_OP_FP BRANCHES
```

```
                                       %      mode register
                                       %      -------------
      PRO.STAA        3;               % PROCESSOR STATE.
      DEBUG.ENA       1;               % DEBUG INTERRUPT ENABLE.
      SOFTFAULTA      1;               % SOFT FAULT.
      PRIVILGEA       1;               % PRIVILEGED MODE.
      TRACE.ENA       1;               % TRACE INTERRUPT ENABLE.
      SNAP.ENA        1;               % SNAP ENABLE.

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %       FPQ1.  EC[380]                                           %
   %       -------------                                            %
   %       CHAIN LENGTH --> 1 BIT.                                  %
   %       LOCATION ------> A8I2.                                   %
   %       SDI PIN -------> N14.                                    %
   %       SDO PIN -------> N13.       PART # : 1995 6002           %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      UNUSED-C        1;               % THIS BIT IS NOT USED.

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %       FPQ1.  EC[379]                                           %
   %       -------------                                            %
   %       CHAIN LENGTH --> 1 BIT.                                  %
   %       LOCATION ------> D2I2.                                   %
   %       SDI PIN -------> N14.                                    %
   %       SDO PIN -------> N13.       PART # : 1995 6002           %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      UNUSED-D        1;               % THIS BIT IS NOT USED.

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %       FPQ1.  EC[378]                                           %
   %       -------------                                            %
   %       CHAIN LENGTH --> 1 BIT.                                  %
   %       LOCATION ------> F6I2.                                   %
   %       SDI PIN -------> N14.                                    %
   %       SDO PIN -------> N13.       PART # : 1995 6002           %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      FBUS1ERR4       1;               % FBUS1 DATA PARITY ERROR.
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %       FFQ1.  EC[377]                                             %
    %       -------------                                              %
    %       CHAIN LENGTH --> 1 BIT.                                    %
    %       LOCATION ------> I0I2.                                     %
    %       SDI PIN -------> N14.                                      %
    %       SDO PIN -------> N13.       PART # : 1995 6002             %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        FBUS1ERR3       1;                  % FBUS1 DATA PARITY ERROR.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %       FFQ1.  EC[376]                                             %
    %       -------------                                              %
    %       CHAIN LENGTH --> 1 BIT.                                    %
    %       LOCATION ------> K4I2.                                     %
    %       SDI PIN -------> N14.                                      %
    %       SDO PIN -------> N13.       PART # : 1995 6002             %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        UNUSED-E        1;                  % THIS BIT IS NOT USED.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %       FFQ1.  EC[375]                                             %
    %       -------------                                              %
    %       CHAIN LENGTH --> 1 BIT.                                    %
    %       LOCATION ------> M8I2.                                     %
    %       SDI PIN -------> N14.                                      %
    %       SDO PIN -------> N13.       PART # : 1995 6002             %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        UNUSED-F        1;                  % THIS BIT IS NOT USED.

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %       FFQ1.  EC[374]                                             %
    %       -------------                                              %
    %       CHAIN LENGTH --> 1 BIT.                                    %
    %       LOCATION ------> P2I2.                                     %
    %       SDI PIN -------> N14.                                      %
    %       SDO PIN -------> N13.       PART # : 1995 6002             %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        UNUSED-G        1;                  % THIS BIT IS NOT USED.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        FRQCTL.  EC[373-317]                                                        %
%        ------------------                                                          %
%        CHAIN LENGTH --> 57 BIT.                                                    %
%        LOCATION ------> K4K6.                                                      %
%        SDI PIN -------> N11.                                                       %
%        SDO PIN -------> NC01        PART # : 1995 9659                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     FUTCURER        1;                  % MCERROR CAUSED BY FETCH OR XM.
     ERRVECTOR       5;                  % MCACM ERROR VECTOR REGISTER.
     XMWRITE         1;                  % TAG WAS FOR AN XM WRITE.
     XMREAD          1;                  % TAG WAS FOR XM READ.
     FAULT           4;                  % FAULT INDICATOR FLAGS 1/PAGE.
     RBUSTAG6        1;                  %<-|
     RBUSTAG5        1;                  %  |
     RBUSTAG4        1;                  %  |
     RBUSTAG3        1;                  %  | RBUS TAG FIELD FROM MCACM.
     RBUSTAG2        1;                  %  |
     RBUSTAG1        1;                  %  |
     RBUSTAG0        1;                  %<-|
     RBQVLD7         1;                  % THE FOLLLOWING 8 F/F'S
     RBQVLD6         1;                  % ARE THE VALID ENTRY FLAGS
     RBQVLD5         1;                  % FOR THE RBUSQ
     RBQVLD4         1;                  %
     RBQVLD3         1;                  %
     RBQVLD2         1;                  %
     RBQVLD1         1;                  %
     RBQVLD0         1;                  %
     RBQADR          3;                  % CURRENT RBUSQ READ ADDRESS.
     OPQBVLD3        1;                  % OPQB PAGE3 VALID FLAG
     OPQAVLD3        1;                  % OPQA PAGE3 VALID FLAG
     OPQBVLD2        1;                  % OPQB PAGE2 VALID FLAG
     OPQAVLD2        1;                  % OPQA PAGE2 VALID FLAG
     OPQBVLD1        1;                  % OPQB PAGE1 VALID FLAG
     OPQAVLD1        1;                  % OPQA PAGE1 VALID FLAG
     OPQBVLD0        1;                  % OPQB PAGE0 VALID FLAG
     OPQAVLD0        1;                  % OPQA PAGE0 VALID FLAG
     BOVRIDE         1;                  % OPQB VALID OVERRIDE.SET AT
                                         % OPLIT BRANCH,RESET IF CORRECT
                                         % DATA IS RECEIVED,BASICALLY THE
                                         % 2ND DATA FOR THE CURRENT OPQB.
```

```
    AOVRIDE         1;              % OPQA VALID OVERRIDE.SET AT
                                    % OPLIT BRANCH,RESET IF CORRECT
                                    % DATA IS RECEIVED,BASICALLY THE
                                    % 2ND DATA FOR THE CURRENT OPQA.
    FPVALID3        1;              % IF SET, FPAGE3 IS VALID.
    DATAHIT3        1;              % IF SET, FPAGE3 OPQ DATA
                                    % MUST BE READ BY THE XM.
    OPTIMAL3        1;              % MICROCODE FLAG INDICATING
                                    % THE OF HAS SHORT OPERANDS.
    FPVALID2        1;              % IF SET, FPAGE2 IS VALID.
    DATAHIT2        1;              % IF SET, FPAGE2 OPQ DATA
                                    % MUST BE READ BY THE XM.
    OPTIMAL2        1;              % SEE OPTIMAL3.
    FPVALID1        1;              % IF SET, FPAGE1 IS VALID.
    DATAHIT1        1;              % IF SET, FPAGE1 OPQ DATA
                                    % MUST BE READ BY THE XM.
    OPTIMAL1        1;              % SEE OPTIMAL3.
    FPVALID0        1;              % IF SET, FPAGE0 IS VALID.
    DATAHIT0        1;              % IF SET, FPAGE0 OPQ DATA
                                    % MUST BE READ BY THE XM.
    OPTIMAL0        1;              % SEE OPTIMAL3.
    XMFPAGE         2;              % XM'S CURRENT FPAGE POINTER.
    FBUSCERR        1;              % FBUS TAG OR ERROR VECTOR
                                    % PARITY ERROR.
    OFREQUEST       1;              % IF SET, 'OF' SEND THE LAST
                                    % FPUS TRANSFER
    FBUSCTLERR      1;              % PARITY ERROR OVER FBUS1/2
                                    % CONTROL LINES,OR FETCH TRYING
                                    % TO WRITE INTO THE CURRENT XM
                                    % PAGE.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       FPQ2.  EC[316]                                                          %
%       --------------                                                          %
%       CHAIN LENGTH --> 1 BIT.                                                 %
%       LOCATION -------> M8K6.                                                 %
%       SDI PIN --------> R09.                                                  %
%       SDO PIN --------> K02.          PART # : 1995 7356                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    FBUS2ERR1       1;              % FBUS2 DATA PARITY ERROR.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       FF02.  EC[315]                                               %
%       -------------                                                %
%       CHAIN LENGTH --> 1 BIT.                                      %
%       LOCATION ------> P2K6.                                       %
%       SDI PIN -------> R09.                                        %
%       SDO PIN -------> K02.       PART # : 1995 7356               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    FBUS2ERRD      1;                   % FBUS2 DATA PARITY ERROR.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       XMSEQ1.  EC[314-287]                                         %
%       -----------------                                            %
%       CHAIN LENGTH --> 28 BIT.                                     %
%       LOCATION ------> D2D4.                                       %
%       SDI PIN -------> N04.                                        %
%       SDO PIN -------> K02.       PART # : 1995 7554               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    STKTOPMD      12;                   % 12 MSE'S OF STKTOP IN SEQ1D.
    UPCMD         12;                   % MICRO-PROGRAM COUNTER.
    MISMCH1D       1;                   % MISCOMPARE BETWEEN SEQ1D AND
                                        % SEQ1C SETS THIS F/F.
    HFILPTMD       1;                   % POINTER TO OLDEST HISTORY FILE
                                        % ENTRY.
    STKPTRMD       2;                   % STACK WRITE POINTER.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       XMSEQ1.  EC[286-259]                                              %
%       -------------------                                               %
%       CHAIN LENGTH --> 28 BIT.                                          %
%       LOCATION ------> IDD4.                                            %
%       SDI PIN -------> N04.                                             %
%       SDO PIN -------> K02.         PART # : 1995 7554                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   STKTOPMC        12;                 % 12 MSB'S OF STKTOP IN SEQ1C.
   UPCMC           12;                 % MICRO-PROGRAM COUNTER.
   MISMCH1C         1;                 % MISCOMPARE BETWEEN SEQ1C AND
                                       % SEQ1B SETS THIS F/F.
   HFILPTMC         1;                 % POINTER TO OLDEST HISTORY FILE
                                       % ENTRY.
   STKPTRMC         2;                 % STACK WRITE POINTER.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       XMSEQ1.  EC[258-231]                                              %
%       -------------------                                               %
%       CHAIN LENGTH --> 28 BIT.                                          %
%       LOCATION ------> K4D4.                                            %
%       SDI PIN -------> N04.                                             %
%       SDO PIN -------> K02.         PART # : 1995 7554                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   STKTOPMB        12;                 % 12 MSB'S OF STKTOP IN SEQ1B.
   UPCMB           12;                 % MICRO-PROGRAM COUNTER.
   MISMCH1B         1;                 % MISCOMPARE BETWEEN SEQ1B AND
                                       % SEQ1A SETS THIS F/F.
   HFILPTMB         1;                 % POINTER TO OLDEST HISTORY FILE
                                       % ENTRY.
   STKPTRMB         2;                 % STACK WRITE POINTER.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       XMSEQ1.  EC[230-203]                                        %
%       -------------------                                         %
%       CHAIN LENGTH --> 28 BIT.                                    %
%       LOCATION ------> P2D4.                                      %
%       SDI PIN -------> N04.                                       %
%       SDO PIN -------> K02.        PART # : 1995 7554             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    STKTOPMA       12;              % 12 MSE'S OF STKTOP IN SEQ1A.
    UPCMA11-07      5;              % MICRO-PROGRAM COUNTER.
    OPVECTOR        7;
    MISMCH1A        1;              % MISCOMPARE BETWEEN SEQ1A AN
                                    % SEQ1D SETS THIS F/F.
    HFILPTMA        1;              % POINTER TO OLDEST HISTORY F
                                    % ENTRY.
    STKPTRMA        2;              % STACK WRITE POINTER.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       XMSEQ2.  EC[202-191]                                        %
%       -------------------                                         %
%       CHAIN LENGTH --> 12 BIT.                                    %
%       LOCATION ------> F6B0.                                      %
%       SDI PIN -------> R02.                                       %
%       SDO PIN -------> J15.        PART # : 1995 7950             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    STKTOPLB        2;              % 2 LSB'S OF STKTOP.
    UPCLB           2;              % MICRO-PROGRAM COUNTER.
    STKOVFLOB       1;              % PUSHING A 5TH ENTRY INTO
                                    % THE STACK SETS OVERFLOW.
    STKUNFLOB       1;              % POPPING AN EMPTY STACK
                                    % SETS UNDERFLOW.
    STKFULLB        1;              % PUSHING THE 4TH ENTRY INTO
                                    % STACK SETS STACKFULL.
    STKMTYB         1;              % STACK EMPTY FLAG.
    MISMCHLB        1;              %
    HFILPTLB        1;              % HISTORY FILE POINTER.
    STKPTRLB        2;              % 2 MS BITS OF THE STACK POINTER
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       XMSEQ2.  EC[190-179]                                           %
%       ------------------                                             %
%       CHAIN LENGTH --> 12 BIT.                                       %
%       LOCATION ------> M8B0.                                         %
%       SDI PIN -------> R02.                                          %
%       SDO PIN -------> J15.        PART # : 1995 7950                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     STKTOPLA        2;                  % EXPLANATION IS IDENTICAL TO
     UPCLA           2;                  % THE SEQ2B SLICE ABOVE.
     STKOVFLOA       1;                  %
     STKUNFLOA       1;                  %
     STKFULLA        1;                  %
     STKMTYA         1;                  %
     MISMCHLA        1;                  %
     HFILFTLA        1;                  %
     STKPTRLA        2;                  %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THE CHAIN FROM HERE ON IS NOT STRAIGHT FORWARD.THE FOLLOWING IS      %
% A LIST OF THE CHAIN FROM BITS 178-0:(CONSISTING OF 5 CREG CHIPS)     %
%                                                                      %
% 178-- LOCATION --> D2B0, LENGTH --> 6, SDI --> A11, SDO --> A02      %
%       LOCATION --> I0B0, LENGTH --> 6, SDI --> A11, SDO --> A02      %
%       LOCATION --> K4B0, LENGTH --> 6, SDI --> A11, SDO --> A02      %
%       LOCATION --> P2B0, LENGTH --> 6, SDI --> A11, SDO --> A02      %
%       LOCATION --> R6I2, LENGTH --> 6, SDI --> A11, SDO --> A02      %
%       LOCATION --> D2B0, LENGTH -->27, SDI --> P04, SDO --> K14      %
%       LOCATION --> I0B0, LENGTH -->27, SDI --> P04, SDO --> K14      %
%       LOCATION --> K4B0, LENGTH -->27, SDI --> P04, SDO --> K14      %
%       LOCATION --> P2B0, LENGTH -->27, SDI --> P04, SDO --> K14      %
%       LOCATION --> R6I2, LENGTH -->27, SDI --> P04, SDO --> K14      %
% 000-- LOCATION --> R6I2, LENGTH -->14, SDI --> N04, SDO --> C01      %
%                                                                      %
%                            CREG PART # : 1995 8057                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
        UNUSED-H        1;                  % EC[178-173].
        SPAREG4         5;                  % SELECT CRAM BIT TO BE REPLACED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        UNUSED-I        1;                  % EC[172-167].
        SPAREG3         5;                  % SELECT CRAM BIT TO BE REPLACED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        UNUSED-J        1;                  % EC[166-161].
        SPAREG2         5;                  % SELECT CRAM BIT TO BE REPLACED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        UNUSED-K        1;                  % EC[160-155].
        SPAREG1         5;                  % SELECT CRAM BIT TO BE REPLACED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        UNUSED-L        1;                  % EC[154-149].
        SPAREG0         5;                  % SELECT CRAM BIT TO BE REPLACED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        CREGSPARE       1;                  % EC[148-14].
        CRAMPARITY      1;
        FPAADR          3;
        OPGEREQ         1;
        CREG128         1;
        RBOFQSEL        1;
        SPAADR          4;
        SPBADR          4;
        SPWE            1;
        SPINSEL2        1;
        SPINSEL1        1;
        SPINSEL0        1;
        SPCADR          4;
        CREG110         1;
        CREG109         1;
        CREG108         1;
        PTESTFUNC       5;
        PTREGLD         1;
        MSOCMPR         1;
        LUFUNC          2;
        LUASEL1         1;
        LUASEL0         1;
        LUBSEL          3;
        CREG93          1;
        LITADR          8;
        LITSEL          1;
        MSWE            1;
        MSSEL           2;
```

```
        MSREGLD          1;
        MSADRSEL1        1;
        MSADRSEL0        1;
        RQCMD            5;
        RQSPADR          2;
        RQDSEL           3;
        RQASEL           3;
        RQLSEL           3;
        PMASEL           3;
        PMBSEL1          1;
        PMBSEL0          1;
        PMAFUNC          4;
        PMBFUNC          4;
        CREG48           1;
        CREG47           1;
        CREG46           1;
        CREG45           1;
        CREG44           1;
        PMREGLD          1;
        CTLOUTSEL        1;
        RBOFQREQ         1;
        DREGLD           1;
        CREG39           1;
        AUASEL           3;
        AUBSEL2-1        2;
        AUBSEL0          1;
        CREG32           1;
        CREG31           1;
        CREG30           1;
        CREG29           1;
        TESTSEL          5;
        INTCMD           6;
        UMUXSEL          3;
        PUSH             1;
        JUMPADR         14;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        CSADR           14;                     % EC[13-0].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    SDI TO XMC.

    CPMEND;                                     % END OF CHAIN.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            END OF CHAINLIST.START REGISTER LIST                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

REGLIST      FC;

    UPC         3;

        UPCMA11-07;                     % FIRST COPY OF THE UPC
        OPVECTOR;                       % COMBINED FROM SEQ1A AND SEQ2A.
        UPCLA #                         %

    CREG        63;

        CREGSPARE;                      % SPARE REGISTER BIT NOT USE
        CRAMPARITY;                     % EVEN PARITY ACROSS CREG
                                        % --------MISCELLANEOUS CONTROL-
        FPAADR;                         % ADDRESS TO FETCH PAGE A PORT.
        OPQBREQ;                        % OPQB DATA REQUEST.
        CREG128;                        % NOT USED
        RBOPQSEL;                       % WHEN SET IT SELECTS RBQ
                                        % --------SP CONTROL-----------
        SPAADR;                         % SPA ADDRESS FIELD
        SPBADR;                         % SPB ADDRESS FIELD
        SPWE;                           % WRITE ENABLE TO SP
        SPINSEL2;                       % ] WRITE DATA INPUT SELECT
        SPINSEL1;                       % ] TO SCRATCHPAD.
        SPINSEL0;                       % ]
        SPCADR;                         % WRITE ADDRESS TO SP.
                                        % --------PTEST CONTROL---------
        CREG110;                        % ] DATA INPUT SELECT TO PTEST.
        CREG109;                        % ]
        CREG108;                        % ]
        PTESTFUNC;                      % PTEST FUNCTION SELECT.
        PTREGLD;                        % PTREG LOAD LINE.
                                        % --------LU CONTROL-----------
        MSDCMPR;                        % MOST SIGNIFICANT DIGIT
                                        % COMPARE IN THE LU.
        LUFUNC;                         % LOGIC UNIT FUNCTION SELECT.
        LUASEL1;                        % SELECT TO LU A-INPUT.
        LUASEL0;                        %
        LUBSEL;                         % SELECT TO LU B-INPUT.
        CREG93;                         % NOT USED.
```

```
                                % --------LIT FILE CONTROL------
        LITADR;                 % LITFILE ADDRESS.
        LITSEL;                 % SELECTS BYTE LITERAL INSTEAD
                                % OF THE LIT FILE.
                                % --------MASS STORE CONTROL----
        MSWE;                   % MASS STORE WRITE ENABLE.
        MSSEL;                  % MASS STORE INPUT SELECT.
        MSREGLD;                % MASS STORE REGISTER LOAD.
        MSADRSEL1;              % ] MASS STORE ADDRESS SELECT.
        MSADRSEL0;              % ]
                                % --------MEM RQSTR CONTROL-----
        RQCMD;                  % REQUESTOR COMMAND FIELD.
        RQSPADR;                % REQUESTOR SP ADDRESS.
        RQDSEL;                 % REQUESTOR DATA SOURCE SELECT.
        RQASEL;                 % REQUESTOR ADDR SOURCE SELECT.
        RQLSEL;                 % REQUESTOR LENGTH SOURCE SELECT
                                % --------PMUX/CONCAT CONTROL---
        PMASEL;                 % PMUXA INPUT SELECT.
        PMBSEL1;                % ] PMUXB INPUT SELECT.
        PMBSEL0;                % ]
        PMAFUNC;                % PMUXA FUNCTION SELECT.
        PMBFUNC;                % PMUXB FUNCTION SELECT.
        CREG48;                 % ] THIS IS THE CONCATENATE
        CREG47;                 % ] FUNCTION FIELD.
        CREG46;                 % ]
        CREG45;                 % ]
        CREG44;                 % ]
        PMREGLD;                % PMREG LOAD.
                                % --------MISCELLANEOUS CONTROL-
        CTLOUTSEL;              % CONTROL OUT MUX SELECT.
        RBOPQREQ;               % RBOPQ REQUEST LINE.
        DREGLD;                 % DREG LOAD LINE.
        CREG39;                 % NOT USED.
                                % --------AU CONTROL------------
        AUASEL;                 % AU A-INPUT SOURCE SELECT.
        AUBSEL2-1;              % AU B-INPUT SOURCE SELECT.
        AUBSEL0;
        CREG32;                 % ] AU FUNCTION SELECT.
        CREG31;                 % ]
        CREG30;                 % ]
        CREG29;                 % ]
```

```
                                    % --------SEQUENCER CONTROL-----
        TESTSEL;                    % TEST CONDITION SOURCE SELECT.
        INTCMD;                     % INTERNAL COMMAND FIELD.
        UMUXSEL;                    % NEXT MICROADDRESS SELECT.
        PUSH;                       % STACK PUSH BIT.
        JUMPADR #                   % BRANCH ADDRESS FIELD.

                                    % THIS REGISTER CONTAINS ALL THE
    XMERROR    32;                  % ERROR INFORMATION IN THE XM.

        XMDEAD;                     %<-------------------------------
        XMD1ERROR;                  %<  INDICATORS ONLY.THEY DO NOT
        XMD0ERROR;                  %<  CAUSE FREEZE.RESETTING THEM
        FMUXBERROR;                 %<  WILL NOT CHANGE ANYTHING.
        FMUXAERROR;                 %<  (THESE ARE COPIES)
        RWAU0ERROR;                 %<
        RWAU1ERROR;                 %<  ALL THESE SET 1 CLOCK AFTER
        REAM0ERROR;                 %<  THE ERROR F/F GETS SET.
        REAM1ERROR;                 %<  CHECK XMD FOR POSSIBLE ERROR
        RCTAGERROR;                 %<
        PTESTERROR;                 %<-------------------------------
        IOMC.ERROR;                 %<
        TIMEOUT.2B;                 %<
        TIMEOUT.2A;                 %<
        TIMERERR;                   %<
        FRQCTERROR;                 %<
        FPQ2ERROR;                  %< IF ANY OF THESE ARE SET CHECK
        FPBERROR;                   %< XMC.(PROBABLE CAUSE)
        FPAERROR;                   %<
        ERRORB;                     %<
        ERRORA;                     %<
        RBUSCERR;                   %<
        MISMCH1D;                   %<
        MISMCH1C;                   %<
        MISMCH1B;                   %<
        MISMCH1A;                   %<
        STKOVFLOB;                  %<
        STKUNFLOB;                  %<
        MISMCHLB;                   %<
        STKOVFLOA;                  %<
        STKUNFLOA;                  %<
        MISMCHLA #                  %<-------------------------------
```

```
FETCHERROR  6;

     FETCHERR;                  % SET BY ANY FETCH RELATED ERROR
     FBUSCTLERR;                % FETCH CONTROL LINE ERROR.
     FBUS1ERR4;                 % FBUS1 DATA ERROR.
     FBUS1ERR3;                 %   "     "     "   .
     FBUS2ERR1;                 % FBUS2 DATA ERROR.
     FBUS2ERR0 #                %   "     "     "   .

MCERROR     4;

    XMWRITE;                    % ERROR CAUSED BY AN XM WRITE
    XMREAD;                     % ERROR CAUSED BY AN XM READ.
    FUTCURER;                   % ERROR CAUSED BY FETCH.
    ERRVECTOR #                 % MCACM ERROR VECTOR.

LIVEERZ     8;

     SOFTSNAPA;                 % SOFT SNAP BIT.
     PRO.HALTA;                 % PROCESSOR HALT.
     FUTURERR;                  % FETCH RELATED ERROR.
     STOPEVENT;                 % STOP LOGIC DETECTED EVENT.
     AWBUSY;                    % AWBUS ACCESS DENIED.
     CREGERR;                   % CREG PARITY ERROR.
     NOFPAGE;                   % NO VALID FETCH PAGE.
     NORDATA #                  % NO VALID RBUSQ/OPQ DATA.

STACKTOP    2;

    STKTOPMA;                   % THESE ARE THE 12 MOST SIG BITS
                                % OF THE STACKTOP FROM SEQ1A.
    STKTOPLA #                  % THESE ARE THE 2 LEAST SIG BITS
                                % OF THE STACKTOP FROM SEQ2A.

FPVALID     4;

    FPVALID3;                   % VALID FLAG FOR EACH FETCH PAGE
    FPVALID2;
    FPVALID1;
    FPVALID0 #
```

```
INTRPREGAA  6;                  % INTERRUPT REGISTER A (SLICE A)

    TIMEOUT.1A;
    TSKTIMFLTA;
    TSKTIMINTA;
    REAL.IOCA;
    ERROR.IOCA;
    NORM.IOCA #

INTRPREGBA  6;                  % INTERRUPT REGISTER B (SLICE A)

    SNAPICTAKA;
    IPCA;
    TRACDEBUGA;
    FAULTA;
    SMC.INTA;
    TIMEOUT.2A #

MODEREGA    6;                  % MODE REGISTER IN SLICE A.

    PRO.STAA;
    DEBUG.ENA;
    SOFTFAULTA;
    PRIVILGEA;
    TRACE.ENA;
    SNAP.ENA #

MISCREGA    7;                  % MISC REGISTER IN SLICE A.

    SYSFLUSHA;
    FETFLUSHA;
    NOTRYA;
    SOFTSNAPA;
    PRO.HALTA;
    MOPOKA;
    LITERAL2A #

INTMASKA    6;                  % INTERRUPT MASK IN SLICE A.

    TIMOUTMSKA;
    TTF.MASKA;
    TTJ.MASKA;
    REAL.MASKA;
    ERR.MASKA;
    NORM.MASKA #
```

```
    INTRPREGAB  6;                  % INTERRUPT REGISTER A (SLICE B)

        TIMEOUT.1B;
        TSKTIMFLTB;
        TSKTIMINTB;
        REAL.IOCB;
        ERROR.IOCB;
        NORM.IOCB #

    INTRPREGBB  6;                  % INTERRUPT REGISTER B (SLICE B)

        SNAPICTAKB;
        IPCB;
        TRACDEBUGB;
        FAULTB;
        SMC.INTB;
        TIMEOUT.2B #

    MODEREGB    6;                  % MODE REGISTER IN SLICE B.

        PRO.STAB;
        DEBUG.ENB;
        SOFTFAULTB;
        PRIVILGEB;
        TRACE.ENB;
        SNAP.ENB #

   MISCREGB     7;                  % MISC REGISTER IN SLICE B.

        SYSFLUSHB;
        FETFLUSHB;
        NOTRYB;
        SOFTSNAPB;
        PRO.HALTB;
        MCPCKB;
        LITERAL2B #

    INTMASKB    6;                  % INTERRUPT MASK IN SLICE B.

        TIMOUTMSKB;
        TTF.MASKB;
        TTI.MASKB;
        REAL.MASKB;
        ERR.MASKB;
        NORM.MASKB #
```

```
        RBUSQVALID  8;

            RBQVLD7;                    % RBUSQ VALID F/F STARTING
            RBQVLD6;                    % WITH THE 8TH ENTRY.
            RBQVLD5;
            RBQVLD4;
            RBQVLD3;
            RBQVLD2;
            RBQVLD1;
            RBQVLD0 #

        OPQAVALID   4;

            OPQAVLD3;                   % VALID F/F FOR THE OPQA ENTRIES
            OPQAVLD2;                   % CORRESPONDING TO EACH FPAGE.
            OPQAVLD1;
            OPQAVLD0 #

        OPQBVALID   4;

            OPQBVLD3;                   % VALID F/F FOR THE OPQB ENTRIES
            OPQBVLD2;                   % CORRESPONDING TO EACH FETCH
            OPQBVLD1;                   % PAGE.
            OPQBVLD0 #

        DATAHIT     4;

            DATAHIT3;                   % DATAHIT FOR EACH FETCH PAGE.
            DATAHIT2;
            DATAHIT1;
            DATAHIT0 #

        OPTIMAL     4;

            OPTIMAL3;                   % OPTIMAL FLAG FOR EACH FETCH
            OPTIMAL2;                   % PAGE.
            OPTIMAL1;
            OPTIMAL0 #

        FBUS2ERROR  2;

            FBUS2ERR1;
            FBUS2ERR0 #
```

```
    SEQ1MISMCH  4;

        MISMCH1D;                   % SEQ1D DETECTED A MISMATCH
        MISMCH1C;                   % SEQ1C DETECTED A MISMATCH
        MISMCH1B;                   % SEQ1B DETECTED A MISMATCH
        MISMCH1A #                  % SEQ1A DETECTED A MISMATCH

    IFBUS1      13;

        SPAADR;                     % THE INTERNAL FBUS1 USES
        SPBADR;                     % BITS FROM THE CREG AS
        SPCADR;                     % SOURCE DATA FOR FBUS1.
        FTESTFUNC;
        MSDCMPR;
        LUFUNC;
        MSADRSEL1;
        LITADR;
        PMAFUNC;
        PMBFUNC;
        CREG48;
        CREG47;
        CREG46 #

    IFPARITY1   1;

        LITSEL #                    % PARITY BIT OVER INTERNAL FBUS1

    IFLOAD1     1;                  % INTERNAL FBUS1 LOAD.

        FMBSEL1 #

    IFBUS2      7;

        LUASEL1;
        RQSPADR;                    % ARE USED AS SOURCE DATA FOR
        RQLSEL;                     % THE INTERNAL FBUS2.
        CREG44;
        SPINSEL2;
        MSADRSEL0;
        CREG29 #

    IFPARITY2   1;

        CREG45 #                    % PARITY OVER INTERNAL FBUS2.
```

```
IFLOAD2      1;

    PMBSEL0 #                    % INTERNAL FBUS2 LOAD.

IFBUSADR     3;

    CREG32;                      % THE INTERNAL FBUSES SHARE
    CREG31;                      % THIS SET OF ADDRESS LINES
    CREG30 #

MSADRSEL     2;                  % MASS STORE ADDRESS SOURCE
    MSADRSEL1;                   % SELECT.
    MSADRSEL0 #

PMBSEL       2;
    PMBSEL1;
    PMBSEL0 #

LUASEL       2;
    LUASEL1;
    LUASEL0 #

SPINSEL      3;

    SPINSEL2;
    SPINSEL1;
    SPINSEL0 #

CONCATFUNC   5;

    CREG48;                      % THIS IS THE CONCATENATE
    CREG47;                      % FUNCTION FIELD FROM CREG.
    CREG46;
    CREG45;
    CREG44 #

AUBSEL       2;

    AUBSEL2-1;
    AUBSEL0 #
```

```
AUFUNC      4;

    CREG32;                          % THIS IS THE AU FUNCTION
    CREG31;                          % FIELD FROM THE CREG
    CREG30;
    CREG29 #

PTESTSEL    3;

    CREG110;
    CREG109;
    CREG108 #

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THE FOLLOWING REGISTERS HAVE BEEN INTRODUCED FOR RAM LOAD/VERIFY %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

CSADR2      5;                       % CONTROL STORE ADDRESS.
                                     % DUPLICATED TO PRESERVE PARITY.
     SPAADR;
     SPBADR;
     SPCADR;
     RQSPADR;
     CSADR #

SPRDADR     2;                       % USED AS SCRATCHPAD READ ADDR
                                     % DURING RAM LOAD/VERIFY.
     SPAADR;                         % DUPLICATED TO PRESERVE PARITY.
     SPBADR #

SPWRADR     2;                       % USED AS SCRATCHPAD WRITE ADDR
                                     % DURING RAM LOAD/VERIFY.
     SPBADR;                         % DUPLICATED TO PRESERVE PARITY.
     SPCADR #

LITRWADR    3;                       % USED AS LIT FILE READ ADDRESS
                                     % DURING RAM LOAD/VERIFY.
     SPAADR;                         % DUPLICATED TO PRESERVE PARITY.
     SPBADR;
     LITADR #
```

```
STACKADR    6;          % USED AS STACK POINTER DURING
                        % RAM LOAD/VERIFY.DUPLICATED
     STKPTRMA;          % SO AS TO AVOID MISMATCHES.
     STKPTRMB;
     STKPTRMC;
     STKPTRMD;
     STKPTRLA;
     STKPTRLB #

IFPADR      2;          % INTERNAL FBUS1/2 PAGE POINTER
                        % DURING WRITE.
     CREG110;
     CREG109 #

IFBUS1NP   14;          % INTERNAL FBUS1 INPUT DATA
                        % WHICH INCLUDES THE PARITY BIT
     LITSEL;            % AT THE MSB POSITION.
     SPAADR;
     SPBADR;
     SPCADR;
     FTESTFUNC;
     MSDCMPR;
     LUFUNC;
     MSADRSEL1;
     LITADR;
     PMAFUNC;
     PMBFUNC;
     CREG48;
     CREG47;
     CREG46 #

IFBUS2NP    8;          % INTERNAL FBUS2 INPUT DATA
                        % WHICH INCLUDES THE PARITY BIT
     CREG45;            % IN THE MSB POSITION.
     LUASEL1;
     RQSPADR;
     RQLSEL;
     CREG44;
     SPINSEL2;
     MSADRSEL0;
     CREG29 #
```

```
    FPAADRX2      2;                     % INTERNAL FBUS READ-ADDRESS
                                         % IS DUPLICATED TO MASK OFF ANY
         RQLSEL;                         % CREG PARITY ERRORS.
         FPAADR #

    IFADRX2       6;                     % INTERNAL FBUS WRITE-ADDRESS
                                         % IS DUPLICATED TO MASK OFF ANY
         CREGSPARE;                      % CREG PARITY ERRORS.
         OPOBREG;
         CREG128;
         CREG32;
         CREG31;
         CREG30 #

    STKFLUS       2;
                                         % FAKE REG THAT INCLUDES THIS
         CREGSPARE;                      % BIT AS THE OVERALL CREG PRTY.
         JUMPADR #                       % WRITE DATA IS SET HERE.

    HFILPTR       6;                     % HISTORY FILE POINTER.

         HFILPTMD;
         HFILPTMC;
         HFILPTMB;
         HFILPTMA;
         HFILPTLB;
         HFILPTLA #

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  4/8/86 CHANGES. NEEDED FOR RBUSQ/CFQ READ & WRITE ROUTINES.            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    RBUSTAG       7;                     % RBUS TAG REGISTER.

        RBUSTAG6;
        RBUSTAG5;
        RBUSTAG4;
        RBUSTAG3;
        RBUSTAG2;
        RBUSTAG1;
        RBUSTAG0 #
```

```
RBUSTAGM    4;                  % RBUS TAG MODULE IDENTIFIER.

    RBUSTAG6;
    RBUSTAG5;
    RBUSTAG4;
    RBUSTAG3 #

RBQIADR     3;                  % THAT PORTION OF THE TAG REG
                                % THAT IDENTIFIES THE RBQ ENTRY.
    RBUSTAG2;
    RBUSTAG1;
    RBUSTAG0 #

OPQPAGE     2;                  % THAT PORTION OF THE TAG REG
                                % THAT IDENTIFIES THE XM PAGE.
    RBUSTAG2;
    RBUSTAG1 #

IFFADRX2    3;                  % INTERNAL FBUS1/2 PAGE ADDRESS
                                % DURING WRITE.(RAM LOAD/VERIFY)
    LUFUNC;                     % DUPLICATED TO PRESERVE PARITY.
    CREG110;
    CREG109 #

JMPADRX2    5;                  % CREG JUMPADR,DUPLICATED TO
                                % PRESERVE PARITY.
    LUFUNC;                     % USED DURING RAM LOAD/VERIFY.
    SPAADR;
    SPBADR;
    SPCADR;
    JUMPADR #

RQREG       2;                  % REQUESTOR REGISTER SELECT.

    RQSPADR;
    MSSEL #

FBUS1ERROR  2;                  % USED ONLY IN PATHTESTS.

    FBUS1ERR4;
    FBUS1ERR3 #

OPVECREG    2;                  % USED IN READING OPLIT IN FPG2.

    OPVECTOR;
    LITERAL2A #
```

```
ITAGREG     4;                  % INTERNAL RBUS TAG REGISTER.

    CREG110;
    CREG109;
    CREG108;
    AUBSEL2-1 #

ITAGPRTY    1;                  % INTERNAL RBUS TAG PARITY.

    AUBSEL0 #

IDATAVALID  1;                  % INTERNAL RBUS DATA VALID.

    CREG31 #

IERRVECTOR  1;                  % INTERNAL RBUS ERROR VECTOR.

    CREG30 #

IRBCTLPRTY  1;                  % INTERNAL RBUS CONTROL PARITY.

    CREG32 #

UPCMA       2;                  % 12 MS BITS OF UPCA.

    UPCMA11-07;
    OPVECTOR #

COPDATAB    4;                  % COPREG COPY B.

    COPDATAB-3;
    COPDATAB-2;
    COPDATAB-1;
    COPDATAB-0 #

UPCB        2;

    UPCMB;                      % 2ND COPY OF THE UPC COMBINED
    UPCLA #                     % FROM SEQ1B AND SEQ2A.

UPCC        2;

    UPCMC;                      % 3RD COPY OF THE UPC COMBINED
    UPCLB #                     % FROM SEQ1C AND SEQ2B.
```

```
UPCD        2;

    UPCMD;                      % 4TH COPY OF THE UPC COMBINED
    UPCLB #                     % FROM SEQ1D AND SEQ2B.

STACKTOPB   2;

    STKTOPMB;                   % THESE ARE THE 12 MOST SIG BITS
                                % OF THE STACKTOP FROM SEQ1B.
    STKTOPLA #                  % THESE ARE THE 2 LEAST SIG BITS
                                % OF THE STACKTOP FROM SEQ2A.

STACKTOPC   2;

    STKTOPMC;                   % THESE ARE THE 12 MOST SIG BITS
                                % OF THE STACKTOP FROM SEQ1C.
    STKTOPLB #                  % THESE ARE THE 2 LEAST SIG BITS
                                % OF THE STACKTOP FROM SEQ2B.

STACKTOPD   2;

    STKTOPMD;                   % THESE ARE THE 12 MOST SIG BITS
                                % OF THE STACKTOP FROM SEQ1D.
    STKTOPLB #                  % THESE ARE THE 2 LEAST SIG BITS
                                % OF THE STACKTOP FROM SEQ2B.

REGEND;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THE FOLLOWING REGISTER IS ADDED TO SPEED UP THE RETRY ALGORITHM.   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      RETRYCHAIN  62;                      % REGISTERS TO BE RESTORED FOR
                                           % RETRY PURPOSES.
          TIMEOUT.1A;
          TSKTIMFLTA;
          TSKTIMINTA;
          REAL.IOCA;
          ERROR.IOCA;
          NORM.IOCA;
          SNAPICTAKA;
          IPCA;
          TRACDEBUGA;
          FAULTA;
          SMC.INTA;
          TIMEOUT.2A;
          PRO.STAA;
          DEBUG.ENA;
          SOFTFAULTA;
          PRIVILGEA;
          TRACE.ENA;
          SNAP.ENA;
          TIMOUTMSKA;
          TTF.MASKA;
          TTI.MASKA;
          REAL.MASKA;
          ERR.MASKA;
          NORM.MASKA;
          TIMEOUT.1B;
          TSKTIMFLTB;
          TSKTIMINTB;
          REAL.IOCB;
          ERROR.IOCB;
          NORM.IOCB;
          SNAPICTAKB;
          IPCB;
          TRACDEBUGB;
          FAULTB;
          SMC.INTB;
          TIMEOUT.2B;
          PRO.STAB;
          DEBUG.ENB;
```

```
        SOFTFAULTB;
        PRIVILGEB;
        TRACE.ENB;
        SNAP.ENB;
        TIMOUTMSKB;
        TTF.MASKB;
        TTI.MASKB;
        REAL.MASKB;
        ERR.MASKB;
        NORM.MASKB;
        TOUTCTR;
        TOUTPRTY;
        TASKPRTY;
        TASKTIMR;
        USECPRTY;
        USECCTR;
        USECSYNC;
        USECSYNCB;
        COVFLOWB;
        COVFLOWA;
        CCOMLB;
        CCOMHB;
        CCOMLA;
        CCOMHA #

    XMERROR1    14;                 % USED BY THE FAULT FINDER.

        TIMERERR;
        FPQ2ERROR;
        FPBERROR;
        FPAERROR;
        MISMCH1D;
        MISMCH1C;
        MISMCH1B;
        MISMCH1A;
        STKOVFLOB;
        STKUNFLOB;
        MISMCHLB;
        STKOVFLOA;
        STKUNFLOA;
        MISMCHLA #
```

```
XMERROR2      10;                 % USED BY THE FAULT FINDER.

      XMD1ERROR;
      XMD0ERROR;
      PMUXBERROR;
      PMUXAERROR;
      RWAU0ERROR;
      RWAU1ERROR;
      REAM0ERROR;
      REAM1ERROR;
      RCTAGERROR;
      PTESTERROR #

XMERROR3      3;                  % USED BY THE FAULT FINDER.

      IOMC.ERROR;
      ERRORA;
      ERRORB #

XMERROR4      4;                  % USED BY THE FAULT FINDER.
                                  % POSSIBLE MICROCODE FAULT.
      STKOVFLOB;
      STKOVFLOA;
      STKUNFLOB;
      STKUNFLOA #

XMERROR5      2;                  % USED BY THE FAULT FINDER.
                                  % POSSIBLE MICROCODE FAULT.
      FPAERROR;
      FPBERROR #

FETCHEROR1    2;                  % USED BY THE FAULT FINDER.

      FETCHERR;
      FRUSCTLERR #

FRU-REG       10;                 % USED BY THE FUNCTION TEST
                                  % DRIVER,TO IDENTIFY FAULTY FRUS
      PMBSEL1;
      PMBSEL0;
      PMAFUNC;
      PMBFUNC;
      CREG48;
      CREG47;
```

```
        CREG46;
        CREG45;
        CREG44;
        PMREGLD #

    TESTNUMBER  6;                      % USED BY THE FUNCTION TEST DRI-
                                        % VER TO IDENTIFY FAILING TEST.
        LUFUNC;
        LUASEL1;
        LUASEL0;
        LUBSEL;
        CREG93;
        LITADR #

    REAL-ERROR  9;                      % USED BY THE FUNCTION TEST DRI-
                                        % VER TO CONFIRM REAL FAILURES.
        SPWE;
        SPINSEL2;
        SPINSEL1;
        SPINSEL0;
        SPCADR;
        CREG110;
        CREG109;
        CREG108;
        PTESTFUNC #

    REGEND;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               END OF CONTROL CHAIN REGISTER LIST                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

XM DATA CARD CHAIN LIST

```
%  MODULE NAME : XMD                    DATE     : 06-24-85          %
%                                                                     %

DATE    J080786;                  %

CHAIN    ED;                      %XM DATA CARD CHAIN,444 BITS LONG.

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %       AU2 MOST SIGNIFICANT SLICE.  ED[441-408]                  %
        %       ----------------------------------------                  %
        %       CHAIN LENGTH --> 34 BITS.                                 %
        %       LOCATION ------> F6K6.                                    %
        %       SDI PIN  ------> C11.                                     %
        %       SDO PIN  ------> P15.          PART # : 1996 0558         %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

          AUM-ERROR        1;      %ERROR INDICATOR FOR AUM SLICE.
          AUM-EQLOL        1;      %AUM=0 FLAG.ACTIVE LOW.
          AU-CARRY         1;      %CARRY OUT OF THE XM AU.
          CARRY-8TO0       9;      %CARRY OUT OF REMAINING DIGITS.
          AUM-AUREG       20;      %MOST SIG. 5 DIGITS OF AUREG.
          AUM-ADD          1;      %ADDITION INDICATOR.
          AUMBINSLCT       1;      %INDICATES BINARY RESULT SELECTION.

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %       AU2 LEAST SIGNIFICANT SLICE.  ED[407-374]                 %
        %       ----------------------------------------                  %
        %       CHAIN LENGTH --> 34 BITS.                                 %
        %       LOCATION ------> M8K6.                                    %
        %       SDI PIN -------> C11.                                     %
        %       SDO PIN -------> P15.          PART # : 1996 0558         %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

          AUL-ERROR        1;      %ERROR INDICATOR FOR AUL SLICE.
          AUL-EQLOL        1;      %AUL=0 FLAG.ACTIVE LOW.
          AUL-CARRY       10;      %CARRY OUT OF EACH DIGIT.
          AUL-AUREG       20;      %LEAST SIG. 5 DIGITS OF AUREG.
          AUL-ADD          1;      %ADDITION INDICATOR.
          AULBINSLCT       1;      %INDICATES BINARY RESULT SELECTION.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       PTEST MOST SIGNIFICANT SLICE.  ED[373-362]                            %
%       ---------------------------------------                               %
%       CHAIN LENGTH --> 12 BITS.                                             %
%       LOCATION ------> 1D12.                                                %
%       SDI PIN -------> P14.                                                 %
%       SDO PIN -------> C01.          PART # : 1995 8156                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  PTRAMWEN-1      1;      %SOFT PTEST RAM WRITE ENABLE.
  PTINMPRTY       1;      %MOST SIG. 20 BIT PARITY ON PTIN.
  PTRAMBIT18      1;      %SOFT PTEST RAM BIT 18.
  PTM-ERROR       1;      %ERROR INDICATOR FOR PTESTM SLICE.
  PTINPRTY1       1;      %PARITY GENERATED OVER PTIN.
  PTRAMPRTY1      1;      %PARITY GENERATED OVER PTRAM DATA.
  PTM-PTREG       4;      %PTEST REGISTER.
  PTREGPRTY1      1;      %PARITY OVER PTREG.
  LFRAMWEN-1      1;      %LIT FILE RAM WRITE ENABLE.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       PTEST LEAST SIGNIFICANT SLICE.  ED[361-350]                           %
%       ---------------------------------------                               %
%       CHAIN LENGTH --> 12 BITS.                                             %
%       LOCATION ------> K4J2.                                                %
%       SDI PIN -------> P14.                                                 %
%       SDO PIN -------> C01.          PART # : 1995 8156                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  PTRAMWEN-0      1;      %SOFT PTEST RAM WRITE ENABLE.
  PTINLPRTY       1;      %LEAST SIG. 20 BIT PARITY ON PTIN.
  PTRAMBIT08      1;      %SOFT PTEST RAM BIT 8.
  PTL-ERROR       1;      %ERROR INDICATOR FOR PTEST L SLICE.
  PTINPRTY0       1;      %PARITY GENERATED OVER PTIN.
  PTRAMPRTY0      1;      %PARITY GENERATED OVER PTRAM DATA.
  PTLPTREG3       1;      %PTEST REGISTER.
  PTLPTREG2       1;      %PTEST REGISTER.
  PTLPTREG1       1;      %PTEST REGISTER.
  PTLPTREG0       1;      %PTEST REGISTER.
  PTREGPRTY0      1;      %PARITY OVER PTREG.
  LFRAMWEN-0      1;      %LIT FILE RAM WRITE ENABLE.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       PARITY MOST SIGNIFICANT SLICE.  ED[349-336]                       %
%       ----------------------------------------                          %
%       CHAIN LENGTH --> 14 BITS.                                         %
%       LOCATION ------> 10F8.                                            %
%       SDI PIN -------> R10.                                             %
%       SDO PIN -------> N13.          PART # : 1995 9055                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  MSREGDIG1         4;      %MASS STORE ADDRESS REG. DIGIT 1.
  SPINMPRTY         1;      %MOST SIG. 20 BIT PARITY ON SPIN.
  FPATOSPIN1        1;      %SPINSEL SELECTS FETCH PAGE A.
  DREGPRTY1         1;      %MOST SIG. 20 BIT PARITY ON DREG.
  PMAIN-ERR         1;      %ERROR INDICATOR ON PMUXA INPUT.
  RBPRTYERR1        1;      %PARITY ERROR ON RBUS DATA FIELD.
  CREGERROR1        1;      %PARITY ERROR IN THE CREG FIELD.
  SPRBQERR1         1;      %PARITY ERROR,WITH SPRBQ AS A
                            %PROBABLE CAUSE.
  ICUERROR1         1;      %PARITY ERROR,WITH ICU AS A
                            %PROBABLE CAUSE.
  LFICUERR1         1;      %PARITY ERROR,WITH EITHER ICU OR
                            %LIT FILE AS A PROBABLE CAUSE.
  FFICUERR1         1;      %PARITY ERROR,WITH EITHER ICU OR
                            %FETCH PAGES AS A PROBABLE CAUSE.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       PARITY LEAST SIGNIFICANT SLICE.  ED[335-322]                      %
%       ----------------------------------------                          %
%       CHAIN LENGTH --> 14 BITS.                                         %
%       LOCATION ------> K4F8.                                            %
%       SDI PIN -------> R10.                                             %
%       SDO PIN -------> N13.          PART # : 1995 9055                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  MSREGDIG0         4;      %MASS STORE ADDRESS REG. DIGIT 0.
  SPINLPRTY         1;      %LEAST SIG. 20 BIT PARITY ON SPIN.
  FPATOSPIN0        1;      %SPINSEL SELECTS FETCH PAGE A.
  DREGPRTY0         1;      %LEAST SIG. 20 BIT PARITY ON DREG.
  PMBIN-ERR         1;      %ERROR INDICATOR ON PMUXB INPUT.
  RBPRTYERR0        1;      %PARITY ERROR ON RBUS DATA FIELD.
  CREGERROR0        1;      %PARITY ERROR IN THE CREG FIELD.
  SPRBQERR0         1;      %PARITY ERROR,WITH SPRBQ AS A
                            %PROBABLE CAUSE.
  ICUERROR0         1;      %PARITY ERROR,WITH ICU AS A
                            %PROBABLE CAUSE.
```

```
    LFICUERRO       1;      %PARITY ERROR,WITH EITHER ICU OR
                            %LIT FILE AS A PROBABLE CAUSE.
    FPICUERRO       1;      %PARITY ERROR,WITH EITHER ICU OR
                            %FETCH PAGES AS A PROBABLE CAUSE.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       MISC2 ARRAY.  ED[321-298]                                            %
%       --------------------------                                           %
%       CHAIN LENGTH --> 24 BITS.                                            %
%       LOCATION ------> I0D4.                                               %
%       SDI PIN -------> B11.                                                %
%       SDO PIN -------> D03.          PART # : 1996 0350                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    PTRAMPRTY       5;      %PARITY OVER PTEST RAMS.
    A-GREATR-B      1;      %A GREATER THAN B FLAG.
    LU-IS-0         1;      %LOGICAL UNIT OUTPUT IS 0.
    AEQUALTOB       1;      %A EQUAL TO B FLAG.
    ECC-ERROR       1;      %INDICATES MSTORE ECC ERROR.
    MSTORE-ECC      7;      %MSTORE ECC BITS DURING READ.
    SYNDROME        7;      %MSTORE ECC ERROR SYNDROME.
    MISCERROR       1;      %MSTORE DATA INPUT ERROR.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       XMPMUX B ARRAY.  ED[297-293]                                         %
%       --------------------------                                           %
%       CHAIN LENGTH --> 5 BITS.                                             %
%       LOCATION ------> K4P0.                                               %
%       SDI PIN -------> B09.                                                %
%       SDO PIN -------> B14.          PART # : 1995 6051                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    PMUXERRORB      1;      %PMUXB INTERNAL ERROR.
    PMREGB          4;      %PMUXB ROTATE FACTOR REGISTER.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       XMPMUX A ARRAY.  ED[292-288]                                         %
%       --------------------------                                           %
%       CHAIN LENGTH --> 5 BITS.                                             %
%       LOCATION ------> I0P0.                                               %
%       SDI PIN -------> B09.                                                %
%       SDO PIN -------> B14.          PART # : 1995 6051                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    PMUXERRORA      1;      %PMUXA INTERNAL ERROR.
    PMREGA          4;      %PMUXA ROTATE FACTOR REGISTER.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 9.  ED[287-283]                                       %
%       ---------------------------------                                     %
%       CHAIN LENGTH --> 5 BITS.                                              %
%       LOCATION ------> A8F8.                                                %
%       SDI PIN -------> A07.                                                 %
%       SDO PIN -------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU9ERROR         1;      %ICU DIGIT 9 INTERNAL ERROR.
  DREG-DIG9         4;      %9TH DIGIT OF DREG.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 8.  ED[282-278]                                       %
%       ---------------------------------                                     %
%       CHAIN LENGTH --> 5 BITS.                                              %
%       LOCATION ------> A8D4.                                                %
%       SDI PIN -------> A07.                                                 %
%       SDO PIN -------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU8ERROR         1;      %ICU DIGIT 8 INTERNAL ERROR.
  DREG-DIG8         4;      %8TH DIGIT OF DREG.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 7.  ED[277-273]                                       %
%       ---------------------------------                                     %
%       CHAIN LENGTH --> 5 BITS.                                              %
%       LOCATION ------> D2D4.                                                %
%       SDI PIN -------> A07.                                                 %
%       SDO PIN -------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU7ERROR         1;      %ICU DIGIT 7 INTERNAL ERROR.
  DREG-DIG7         4;      %7TH DIGIT OF DREG.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 6.  ED[272-268]                                       %
%       ---------------------------------                                     %
%       CHAIN LENGTH --> 5 BITS.                                              %
%       LOCATION ------> D2E0.                                                %
%       SDI PIN -------> A07.                                                 %
%       SDO PIN -------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU6ERROR         1;      %ICU DIGIT 6 INTERNAL ERROR.
  DREG-DIG6         4;      %6TH DIGIT OF DREG.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 5.  ED[267-263]                                       %
%       ---------------------------------                                     %
%       CHAIN LENGTH --> 5 BITS.                                              %
%       LOCATION ------> D2F8.                                                %
%       SDI PIN -------> A07.                                                 %
%       SDO PIN -------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU5ERROR         1;      %ICU DIGIT 5 INTERNAL ERROR.
  DREG-DIG5         4;      %5TH DIGIT OF DREG.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 4.  ED[262-258]                                       %
%       ---------------------------------                                     %
%       CHAIN LENGTH --> 5 BITS.                                              %
%       LOCATION ------> D2F8.                                                %
%       SDI PIN -------> A07.                                                 %
%       SDO PIN -------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU4ERROR         1;      %ICU DIGIT 4 INTERNAL ERROR.
  DREG-DIG4         4;      %4TH DIGIT OF DREG.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 3.  ED[257-253]                                       %
%       ---------------------------------                                     %
%       CHAIN LENGTH --> 5 BITS.                                              %
%       LOCATION ------> R2E0.                                                %
%       SDI PIN -------> A07.                                                 %
%       SDO PIN -------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU3ERROR         1;      %ICU DIGIT 3 INTERNAL ERROR.
  DREG-DIG3         4;      %3RD DIGIT OF DREG.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 2.  ED[252-248]                                        %
%       -------------------------------                                        %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION -------> R6B0.                                                %
%       SDI PIN --------> A07.                                                 %
%       SDO PIN --------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU2ERROR         1;      %ICU DIGIT 2 INTERNAL ERROR.
  DREG-DIG2         4;      %2ND DIGIT OF DREG.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 1.  ED[247-243]                                        %
%       -------------------------------                                        %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION -------> R6D4.                                                %
%       SDI PIN --------> A07.                                                 %
%       SDO PIN --------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU1ERROR         1;      %ICU DIGIT 1 INTERNAL ERROR.
  DREG-DIG1         4;      %1ST DIGIT OF DREG.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       ICU DIGIT SLICE 0.  ED[242-238]                                        %
%       -------------------------------                                        %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION -------> P2D4.                                                %
%       SDI PIN --------> A07.                                                 %
%       SDO PIN --------> F02.          PART # : 1995 8602                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ICU0ERROR         1;      %ICU DIGIT 0 INTERNAL ERROR.
  DREG-DIG0         4;      %0TH DIGIT OF DREG.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPRBQ DIGIT SLICE 9.  ED[237-233]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION ------> A8N0.                                                 %
%       SDI PIN -------> R09.                                                  %
%       SDO PIN -------> K02.          PART # : 1995 8800                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN9PRTY        1;      %PARITY OVER SPIN DIGIT 9.
  MOPREG39         1;      % SPARE REGISTER
  MOPREG38         1;
  MOPREG37         1;
  MOPREG36         1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPRBQ DIGIT SLICE 8.  ED[232-228]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION ------> A8K6.                                                 %
%       SDI PIN -------> R09.                                                  %
%       SDO PIN -------> K02.          PART # : 1995 8800                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN8PRTY        1;      %PARITY OVER SPIN DIGIT 8.
  MOPREG35         1;      % SPARE REGISTER
  MOPREG34         1;
  MOPREG33         1;
  MOPREG32         1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPRBQ DIGIT SLICE 7.  ED[227-223]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION ------> A812.                                                 %
%       SDI PIN -------> R09.                                                  %
%       SDO PIN -------> K02.          PART # : 1995 8800                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN7PRTY        1;      %PARITY OVER SPIN DIGIT 7.
  MOPDIGIT7        4;      %7TH DIGIT OF MOP.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPREG DIGIT SLICE 6.  ED[222-218]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION ------> D212.                                                 %
%       SDI PIN -------> R09.                                                  %
%       SDO PIN -------> K02.          PART # : 1995 8800                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN6PRTY        1;      %PARITY OVER SPIN DIGIT 6.
  MOPDIGIT6        4;      %6TH DIGIT OF MOP.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPREG DIGIT SLICE 5.  ED[217-213]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION ------> D2K6.                                                 %
%       SDI PIN -------> R09.                                                  %
%       SDO PIN -------> K02.          PART # : 1995 8800                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN5PRTY        1;      %PARITY OVER SPIN DIGIT 5.
  MOPDIGIT5        4;      %5TH DIGIT OF MOP.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPREG DIGIT SLICE 4.  ED[212-208]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION ------> P2K6.                                                 %
%       SDI PIN -------> R09.                                                  %
%       SDO PIN -------> K02.          PART # : 1995 8800                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN4PRTY        1;      %PARITY OVER SPIN DIGIT 4.
  MOPDIGIT4        4;      %4TH DIGIT OF MOP.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPREQ DIGIT SLICE 3.  ED[207-203]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION -------> P2I2.                                                %
%       SDI PIN --------> R09.                                                 %
%       SDO PIN --------> K02.          PART # : 1995 8800                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN3PRTY         1;      %PARITY OVER SPIN DIGIT 3.
  MOPDIGIT3         4;      %3RD DIGIT OF MOP.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPREQ DIGIT SLICE 2.  ED[202-198]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION -------> P6I2.                                                %
%       SDI PIN --------> R09.                                                 %
%       SDO PIN --------> K02.          PART # : 1995 8800                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN2PRTY         1;      %PARITY OVER SPIN DIGIT 2.
  MOPDIGIT2         4;      %2ND DIGIT OF MOP.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPREQ DIGIT SLICE 1.  ED[197-193]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION -------> R6K6.                                                %
%       SDI PIN --------> R09.                                                 %
%       SDO PIN --------> K02.          PART # : 1995 8800                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN1PRTY         1;      %PARITY OVER SPIN DIGIT 1.
  MOPDIGIT1         4;      %1ST DIGIT OF MOP

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       SPREQ DIGIT SLICE 0.  ED[192-188]                                      %
%       ---------------------------------                                      %
%       CHAIN LENGTH --> 5 BITS.                                               %
%       LOCATION -------> R6N0.                                                %
%       SDI PIN --------> R09.                                                 %
%       SDO PIN --------> K02.          PART # : 1995 8800                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SPIN0PRTY         1;      %PARITY OVER SPIN DIGIT 0.
  MOPDIGIT0         4;      %0TH DIGIT OF MOP.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       REWAU MOST SIGNIFICANT SLICE.  ED[187-146]          %
%       ------------------------------------------          %
%       CHAIN LENGTH --> 42 BITS.                           %
%       LOCATION ------> F6N0.                              %
%       SDI PIN -------> R14.                               %
%       SDO PIN -------> N03.        PART # : 1995 8859     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  AWDATPTY1        1;      %20 BIT PARITY OVER AWDATA.
  REWAUEN1         1;      %ENABLES REWAU OPERATIONS.
  REWAUERR1        1;      %INTERNAL REWAU ERROR.
  TESTMODE1        1;      %ENABLES MEM RQSTR TESTING.
  RREG35-34        2;      %2 MOST SIG. BITS OF EACH DIGIT
  RREG31-30        2;      %OF THE RESIDUE REGISTER.
  RREG27-26        2;      % THERE ARE A TOTAL OF 9 DIGITS
  RREG23-22        2;      % MAKING THE TOTAL NUMBER OF BITS
  RREG19-18        2;      % EQUAL TO 2X9=18.
  RREG15-14        2;      %THE NUMBERS IN THE NAMES INDICATE
  RREG11-10        2;      %THE BIT POSITION IN THE 36 BIT
  RREG07-06        2;      %RREG REGISTER.
  RREG03-02        2;      %
  AWDAT39-38       2;      %2 MOST SIG. BITS OF EACH DIGIT
  AWDAT35-34       2;      %OF THE AWBUS DATA REGISTER.
  AWDAT31-30       2;      % THERE ARE A TOTAL OF 10 DATA
  AWDAT27-26       2;      % DIGITS,MAKING THE TOTAL NUMBER
  AWDAT23-22       2;      % OF BITS EQUAL TO 2X10=20.
  AWDAT19-18       2;      %
  AWDAT15-14       2;      % THE NUMBERS IN THE NAMES,INDI-
  AWDAT11-10       2;      % CATE THE BIT POSITION IN THE
  AWDAT07-06       2;      % 40 BIT AWDATAREG REGISTER.
  AWDAT03-02       2;      %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       REWAU LEAST SIGNIFICANT SLICE.  ED[145-104]                       %
%       ----------------------------------------                          %
%       CHAIN LENGTH --> 42 BITS.                                         %
%       LOCATION ------> M8N0.                                            %
%       SDI PIN -------> R14.                                             %
%       SDO PIN -------> N03.          PART # : 1995 8859                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  AWDATPTY0        1;      %20 BIT PARITY OVER AWDATA.
  REWAUEN0         1;      %ENABLES REWAU OPERATIONS.
  REWAUERR0        1;      %INTERNAL REWAU ERROR.
  TESTMODE0        1;      %ENABLES MEM RGSTR TESTING.
  RREG33-32        2;      %2 LEAST SIG. BITS OF EACH DIGIT
  RREG29-28        2;      %OF THE RESIDUE REGISTER.
  RREG25-24        2;      %
  RREG21-20        2;      %
  RREG17-16        2;      %
  RREG13-12        2;      %
  RREG09-08        2;      %
  RREG05-04        2;      %
  RREG01-00        2;      %
  AWDAT37-36       2;      %2 LEAST SIG. BITS OF EACH DIGIT
  AWDAT33-32       2;      %OF THE AWBUS DATA REGISTER.
  AWDAT29-28       2;      %
  AWDAT25-24       2;      %
  AWDAT21-20       2;      %
  AWDAT17-16       2;      %
  AWDAT13-12       2;      %
  AWDAT09-08       2;      %
  AWDAT05-04       2;      %
  AWDAT01-00       2;      %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       REAM MOST SIGNIFICANT SLICE.  ED[103-81]                          %
%       ----------------------------------------                          %
%       CHAIN LENGTH --> 23 BITS.                                         %
%       LOCATION ------> D2N0.                                            %
%       SDI PIN -------> A02.                                             %
%       SDO PIN -------> A14.          PART # : 1995 8909                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  LPCTRIS0-1       1;      %LOOP COUNTER STATUS.
  REAMERR1         1;      %INTERNAL REAM ERROR.
  AWADRPTY1        1;      %20 BIT PARITY OVER AWADDR.
  AWADRREG1       20;      %20 MOST SIG. BITS OF AW ADDRESS.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      REAM LEAST SIGNIFICANT SLICE.  ED[80-58]                 %
%      ------------------------------------                     %
%      CHAIN LENGTH --> 23 BITS.                                %
%      LOCATION ------> P2N0.                                   %
%      SDI PIN -------> A02.                                    %
%      SDO PIN -------> A14.       PART # : 1995 8909           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   UNUSED-BIT       1;     %THIS BIT IS ALWAYS 0.
   REAMERR0         1;     %INTERNAL REAM ERROR.
   AWADRPTY0        1;     %20 BIT PARITY OVER AWADDR.
   AWADRREG0       20;     %20 LEAST SIG. BITS OF AW ADDRESS.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      RELM2 ARRAY.  ED[57-36]                                  %
%      --------------------                                     %
%      CHAIN LENGTH --> 22 BITS.                                %
%      LOCATION ------> K4N0.                                   %
%      SDI PIN -------> A08.                                    %
%      SDO PIN -------> D03.       PART # : 1996 0954           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   ROSTRIDLE        1;     %MEM ROSTR IS IDLE.
   RESIDUE          1;     %INDICATES THE EXISTANCE OF RESI-
                           %DUAL DATA IN RREG.
   RELMERROR        1;     %RELM INTERNAL ERROR.
   AWCTLPRTY        1;     %PARITY OVER AW CMD & AW LENGTH.
   WTHRUREG         4;     %STORES THE LENGTH FOR THE 2ND
                           %CYCLE IN CASE OF A WRITE THRU.
   AFREG            4;     %ALIGNEMENT FACTOR REGISTER.
   PHYSLENRD        1;     %ENABLES PHYSICAL LENGTH READ.
                           %(MAINTENANCE MODE.)
   LREG             4;     %LOOP COUNTER INC/DEC FACTOR.
   AWLENGTH         4;     %AW LENGTH REGISTER.
   WLCMD1           1;     %INDICATES THE USAGE OF THE 1ST
                           %SUBROUTINE IN THE WRITE LAST ALG.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       RECTAG2 ARRAY.  FD[35-0]                          %
%       ----------------------                            %
%       CHAIN LENGTH --> 36 BITS.                         %
%       LOCATION ------> TONO.                            %
%       SDI PIN -------> R07.                             %
%       SDO PIN -------> H02.       PART # : 1996 1051    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    OPCOMPLT        1;      %OP COMPLETE FLAG SENT TO FETCH.
    RECTAGERR       1;      %RECTAG INTERNAL ERROR.
    FETFLUSH        1;      %FETCH FLUSH FLAG.
    SYSFLUSH        1;      %SYSTEM FLUSH FLAG.
    LOGICLENRD      1;      %ENABLES LOGICAL LENGTH READ.
                            %(MAINTENANCE MODE.)
    FPINVALID       1;      %FETCH PAGE INVALID FLAG.
    RBQFULL         1;      %RBUS QUEUE FULL FLAG.
    RBQEMPTY        1;      %RBUS QUEUE EMPTY FLAG.
    RQSTCNTR        4;      %RBUS QUEUE REQUEST COUNTER.
    AWTAGPRTY       1;      %AW BUS TAG PARITY.
    AWTAG-6TO3      4;      %AW BUS TAG REGISTER BITS 6,5,4,3.
    FLUSHPAGE       2;      %AW BUS TAG BITS 2,1. THIS IS THE
    AWTAG-0         1;      %PAGE THAT ISSUED THE FLUSH. BIT 0.
    TAGCNTR         3;      %RBUS QUEUE TAG COUNTER.
    LENSTATUS       1;      %LOGICAL LENGTH STATUS.
    DISAWADR        1;      %AW ADDRESS DISABLE FLAG.
    FAULTOVR        1;      %OVERRIDE FAULT.
    REGSEL          2;      %REGISTER SELECT FOR THE 2ND CYCLE.
    RQSTRBUSY       1;      %2ND CYCLE INDICATOR.
    AWCOMMAND       5;      %AW BUS COMMAND REGISTER.
    AWRQST          1;      %INDICATES AN AW BUS REQUEST.
    QFULL           1;      %AW BUS QUEUE FULL FLAG.

CHNEND;                     %END OF CHAIN.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           END OF CHAIN LIST.START REGISTER LIST.              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

REGLIST  ED;

     AUREG            2;         %ARITHMETIC UNIT PIPELINE REGISTER.
                                 %THIS IS A 40 BIT REGISTER.
       AUM-AUREG;
       AUL-AUREG #

     PTRAMDATA       18;         %THIS REGISTER IS USED TO READ PTEST
                                 %RAM DATA.REFER TO XM SPEC.
       PTRAMPRTY;                % 5-BITS OF PARITY ARE INCLUDED
       PTINMPRTY;                % (USED DURING RAM VERIFY).
       PTRAMBIT18;
       PTM-ERROR;
       PTINPRTY1;
       PTRAMPRTY1;
       PTM-PTREG;
       MOFREG39;
       PTINLPRTY;
       PTRAMBIT08;
       PTL-ERROR;
       PTINPRTY0;
       PTRAMPRTY0;
       PTLPTREG3;
       PTLPTREG2;
       PTLPTREG1;
       PTLPTREG0;
       MOFREG35 #

     MSREG            2;         %THIS IS MASS STORE ADDRESS REGISTER.

       MSREGDIG1;
       MSREGDIG0 #
```

```
DREG            10;     %DREG WITHOUT THE PARITY BITS.

  DREG-DIG9;
  DREG-DIG8;
  DREG-DIG7;
  DREG-DIG6;
  DREG-DIG5;
  DREG-DIG4;
  DREG-DIG3;
  DREG-DIG2;
  DREG-DIG1;
  DREG-DIG0 #

MOPREG           8;

  MOPDIGIT7;
  MOPDIGIT6;
  MOPDIGIT5;
  MOPDIGIT4;
  MOPDIGIT3;
  MOPDIGIT2;
  MOPDIGIT1;
  MOPDIGIT0 #

RREG            18;     %THIS REGISTER HOLDS RESIDUAL DATA
                        %IN CASE OF A WRITE ALIGNEMENT.

  RREG35-34;
  RREG33-32;
  RREG31-30;
  RREG29-28;
  RREG27-26;
  RREG25-24;
  RREG23-22;
  RREG21-20;
  RREG19-18;
  RREG17-16;
  RREG15-14;
  RREG13-12;
  RREG11-10;
  RREG09-08;
  RREG07-06;
  RREG05-04;
  RREG03-02;
  RREG01-00 #
```

```
AWDATREG        20;     %AW DATA REGISTER.

  AWDAT39-38;
  AWDAT37-36;
  AWDAT35-34;
  AWDAT33-32;
  AWDAT31-30;
  AWDAT29-28;
  AWDAT27-26;
  AWDAT25-24;
  AWDAT23-22;
  AWDAT21-20;
  AWDAT19-18;
  AWDAT17-16;
  AWDAT15-14;
  AWDAT13-12;
  AWDAT11-10;
  AWDAT09-08;
  AWDAT07-06;
  AWDAT05-04;
  AWDAT03-02;
  AWDAT01-00 #

AWADRREG        2;      %AW ADDRESS REGISTER.

  AWADRREG1;
  AWADRREG0 #
```

```
XMDERROR        38;     %THIS REGISTER CONTAINS ALL THE ERROR
                        %INFORMATION IN THE XM DATA CARD.
  AUM-ERROR;
  AUL-ERROR;
  PTM-ERROR;
  PTL-ERROR;
  PMAIN-ERR;
  PMBIN-ERR;
  RPPRTYERR1;
  RBPRTYERR0;
  CREGERROR1;
  CREGERROR0;
  SPRBQERR1;
  SPRBQERR0;
  ICUERROR1;
  ICUERROR0;
  LFICUERR1;
  LFICUERR0;
  FPICUERR1;
  FPICUERR0;
  ECC-ERROR;
  MISCERROR;
  PMUXERRORA;
  PMUXERRORB;
  ICU9ERROR;
  ICU8ERROR;
  ICU7ERROR;
  ICU6ERROR;
  ICU5ERROR;
  ICU4ERROR;
  ICU3ERROR;
  ICU2ERROR;
  ICU1ERROR;
  ICU0ERROR;
  REWAUERR1;
  REWAUERR0;
  REAMERR1;
  REAMERR0;
  RELMERROR;
  RECTAGERR #
```

```
XMDERROR1       16;     % ERROR REGISTER USED BYE THE FAULT
                        % FINDER PROGRAM.

  AUM-ERROR;
  AUL-ERROR;
  PTM-ERROR;
  PTL-ERROR;
  PMAIN-ERR;
  PMPIN-ERR;
  SPRBQERR1;
  SPRBQERR0;
  ICUERROR1;
  ICUERROR0;
  LFICUERR1;
  LFICUERR0;
  ECC-ERROR;
  MISCERROR;
  ICU9ERROR;
  ICU8ERROR #

XMDERROR2       16;     % ERROR REGISTER USED BYE THE FAULT
                        % FINDER PROGRAM.

  ICU7ERROR;
  ICU6ERROR;
  ICU5ERROR;
  ICU4ERROR;
  ICU3ERROR;
  ICU2ERROR;
  ICU1ERROR;
  ICU0ERROR;
  PMUXERRORA;
  PMUXERRORB;
  REWAUERR1;
  REWAUERR0;
  REAMERR1;
  REAMERR0;
  RELMERROR;
  RECTAGERR #

XMDERROR3        2;     % USED BY THE FAULT FINDER PROGRAM.

  CREGERROR0;
  CREGERROR1 #

XMDERROR4        2;     % USED BY THE FAULT FINDER PROGRAM.

  FFICUERR1;
  FFICUERR0 #
```

```
DREGPLUS        12;     % DREG WITH IT'S 2 PARITY BITS
                        % USED MAINLY DURING RAM LOAD/VER.
  DREGPRTY1;            % MOST SIG. 20 BIT PARITY.
  DREGPRTY0;            % LEAST SIG. 20 BIT PARITY.
  DREG-DIG9;
  DREG-DIG8;
  DREG-DIG7;
  DREG-DIG6;
  DREG-DIG5;
  DREG-DIG4;
  DREG-DIG3;
  DREG-DIG2;
  DREG-DIG1;
  DREG-DIG0 #

PTF0DATA        10;     % SOFT PTEST FUNCTION 0 DATA IN.
                        % (USED DURING RAM LOAD)
  MOPREG36;             % PARITY OVER DREG-DIG4.
  MOPREG35;             %   "      "     "  -DIG3.
  MOPREG34;             %   "      "     "  -DIG2.
  MOPREG33;             %   "      "     "  -DIG1.
  MOPREG32;             %   "      "     "  -DIG0.
  DREG-DIG4;            % FUNCTION 0 DATA(19:16).
  DREG-DIG3;            %    "      "    " (15:12).
  DREG-DIG2;            %    "      "    " (11:08).
  DREG-DIG1;            %    "      "    " (07:04).
  DREG-DIG0 #           %    "      "    " (03:00).

PTF1DATA        10;     % SOFT PTEST FUNCTION 1 DATA IN.
                        % (USED DURING RAM LOAD)
  MOPREG36;             % PARITY OVER DREG-DIG9.
  MOPREG35;             %   "      "     "  -DIG8.
  MOPREG34;             %   "      "     "  -DIG7.
  MOPREG33;             %   "      "     "  -DIG6.
  MOPREG32;             %   "      "     "  -DIG5.
  DREG-DIG9;            % FUNCTION 1 DATA(19:16).
  DREG-DIG8;            %    "      "    " (15:12).
  DREG-DIG7;            %    "      "    " (11:08).
  DREG-DIG6;            %    "      "    " (07:04).
  DREG-DIG5 #           %    "      "    " (03:00).
```

```
BYTELIT          8;      % BYTE LITERAL DATA IN REGISTER.

  MOPREG39;              % (NOT USED CURRENTLY).
  MOPREG38;
  MOPREG37;
  MOPREG36;
  MOPREG35;
  MOPREG34;
  MOPREG33;
  MOPREG32 #

BYTELITP         9;      % BYTE LITERAL DATA IN REGISTER.

  PTLPTREG0;             % BYTELIT PARITY.
  MOPREG39;              %    "    BIT-7
  MOPREG38;              %    "    BIT-6
  MOPREG37;              %    "    BIT-5
  MOPREG36;              %    "    BIT-4
  MOPREG35;              %    "    BIT-3
  MOPREG34;              %    "    BIT-2
  MOPREG33;              %    "    BIT-1
  MOPREG32 #             %    "    BIT-0

PTL-PTREG        4;      % PTREG (COPY 1)

  PTLPTREG3;
  PTLPTREG2;
  PTLPTREG1;
  PTLPTREG0 #

LFILEIN         12;      % LITFILE DATA IN.

  PTLPTREG2;             % MOST SIGNIFICANT 20 BIT PARITY.
  PTLPTREG1;             % LEAST SIGNIFICANT 20 BIT PARITY.
  DREG-DIG9;             % (USED DURING LITFILE RAM LOAD)
  DREG-DIG8;
  DREG-DIG7;
  DREG-DIG6;
  DREG-DIG5;
  DREG-DIG4;
  DREG-DIG3;
  DREG-DIG2;
  DREG-DIG1;
  DREG-DIG0 #
```

```
BYTLOUT          3;      % BYTE LITERAL DATA OUT REGISTER
                         % AND THE ASSOCIATED PARITY.
  DREGPRTYC;             % (USED DURING RAM LOAD/VERIFY)
  DREG-DIG1;
  DREG-DIG0 #

AUM-CARRY        2;      % ALL THE CARRY BITS IN THE AUM
                         % SLICE.(USED IN PATHTESTS)
  AU-CARRY;
  CARRY-8TO0 #

PHYSLEN          4;      % REGISTER USED TO READ THE CONTENT
                         % OF THE PHYSICAL LENGTH RAM IN THE
  RESIDUE;               % MEMORY REQUESTOR.
  AWCTLPRTY;             % (USED FOR MAINTENANCE PURPOSES.)
  AFREG;
  LREG #

LOGICLEN         6;        % REGISTER USED TO READ THE CONTE
                           % OF THE LOGICAL LENGTH RAM IN THE
  AWCOMMAND;               % MEMORY REQUESTOR.
  OPCOMPLT;                % (USED FOR MAINTENANCE PURPOSES.)
  RECTAGERR;
  FETFLUSH;
  SYSFLUSH;
  FPINVALID #
```

```
        AWADDRESS        16;              % USED IN READING THE AW ADDRESS
                                          % RAM CONTENTS.
          MOPREG39;                       % (MAINTENANCE ONLY)
          MOPREG38;
          MOPREG37;
          MOPREG36;
          MOPREG35;
          MOPREG34;
          MOPREG33;
          MOPREG32;
          MOPDIGIT7;
          MOPDIGIT6;
          MOPDIGIT5;
          MOPDIGIT4;
          MOPDIGIT3;
          MOPDIGIT2;
          MOPDIGIT1;
          MOPDIGIT0 #

    REGEND;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                 END OF DATA CHAIN REGISTER LIST.                 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

XM CONTROL CARD CHAIN LIST (MAINTENANCE)

```
   MODULE NAME : XMC                         DATE      : 5/23/85

  DATE     JU12387;                 %

  CHAIN     YEC;                    %XM CONTROL CARD MAINTENANCE CHAIN
                                    % 112 BITS LONG.

       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       %       CLKMNT2 ARRAY.  YEC[111-96]                          %
       %       ---------------------------                          %
       %       CHAIN LENGTH --> 16 BITS.                            %
       %       LOCATION ------> R6F8.                               %
       %       SDI PIN -------> P08.                                %
       %       SDO PIN -------> L13.        PART # : 1996 0053      %
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

         ERRORIG            1;       % error ignore
         OVERRUN            1;       % over run
         COUNTERFF          1;       % counter f/f
         CLKBAD             1;       % clock bad
         SKEWLOW            1;       % skew low f/f
         SKEWHI             1;       % skew hi f/f
         MODIFYSKEW         1;       % modify skew f/f
         SKEWREG            9;       % skew register

       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       %       STOP ARRAY.  YEC[95-0]                               %
       %       ----------------------                               %
       %       CHAIN LENGTH --> 96 BITS.                            %
       %       LOCATION ------> K4N0.                               %
       %       SDI PIN -------> C15.                                %
       %       SDO PIN -------> H14.        PART # : 1995 7307      %
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

         AND.FF             1;       % AND STOP FF.
         OR.FF              1;       % OR STOP FF.
         PRETRIGGER         1;       % PRETRIGGER.
         UNUSED.F           1;       % not used.
         FPAGEMARK          1;       % current PC is marked for a STOP.
         UNUSED.D           2;       % not used
         AN.UNUSEDB         1;       %     "
         OR.UNUSEDB         1;       %     "
```

```
AN.TSK.INT        1;      % AND task timer interrupt to STOP
OR.TSK.INT        1;      % OR task timer interrupt to STOP.
AN.SOFTHLT        1;      % AND soft halt to STOP.
OR.SOFTHLT        1;      % OR soft halt to STOP.
AND.INTRPT        1;      % AND interrupt to STOP.
OR.INTRPT         1;      % OR interrupt to STOP.
ANNONFATAL        1;      % AND nonfatal error to STOP.
ORNONFATAL        1;      % OR nonfatal error to STOP.
AND.FAULT         1;      % AND fault to STOP.
OR.FAULT          1;      % OR fault to STOP.
ANTIMEOUT1        1;      % AND timeout1 to STOP.
ORTIMEOUT1        1;      % OR timeout1 to STOP.
AN.REL.IOC        1;      % AND real time IO complete to STO
OR.REL.IOC        1;      % OR real time IO complete to STOP
AN.ERR.IOC        1;      % AND error IO complete to STOP.
OR.ERR.IOC        1;      % OR error IO complete to STOP.
AN.NOR.IOC        1;      % AND normal IO complete to STOP.
OR.NOR.IOC        1;      % OR normal IO complete to STOP.
AN.EX.TRIG        1;      % AND EXT trigger to stop
OR.EX.TRIG        1;      % OR EXT trigger to stop.
ANSYSFLUSH        1;      % AND system flush to STOP.
ORSYSFLUSH        1;      % OR system flush to STOP.
ANFECHFLUS        1;      % AND fetch flush to STOP.
ORFECHFLUS        1;      % OR fetch flush to STOP.
AND.POP.FP        1;      % AND pop fetch page to STOP.
OR.POP.FP         1;      % OR pop fetch page to STOP.
                          % OUTPUTS:
ERROR             1;      % fake error
INSTRSTOP         1;      % INSTRUCTION STOP indicator.
STOPOREN          1;      % freeze control stop or enable.
STOPANDEN         1;      % freeze control stop and enable.
SOFT.HALT         1;      % soft halt.
UNUSED.E          1;      % not used.
OFF.LINE          1;      % off line.m flush to STOF
TEST.MODE         1;      % test mode.
UNUSED.C          3;      % not used.
SMC.INT           1;      % SMC interrupt.
UNUSED.B          1;      % not used.
HOLD              1;      % hold comparison data.
EUPC             14;      % EUPC comparison data.
UNUSED.A          5;      % not used.
PROSTA-2          1;      % proc. state comparison data bit2
PROSTA-1          1;      %                    "          1
PROSTA-0          1;      %                    "          0
FALSE.E          10;      % not used.
```

```
            ENCOMPUPC       1;      %  UPC comparison enable bit.
            FALSE.C         2;      %     "    .
            PROSTASL-2      1;      % enable stop on proc. state bit 2
            PROSTASL-1      1;      %                    "           1
            PROSTASL-0      1;      %                    "           0
            BPSTOPEN        1;      % backplane STOP enable.
            LOCALEN         1;      % local STOP enable.
            NOT.PROSTA      1;      % stop on NOT of processor state.
            OR.EUPC         1;      % OR eupc to STOP.
            OR.PROSTA       1;      % OR processor state to STOP.
            FALSE.B         1;      % not used.
            AND.EUPC        1;      % AND eupc to STOP.
            AND.PROSTA      1;      % AND procesor state to STOP.
            FALSE.A         1;      % not used

    CHNEND;                         % end of chain.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             END OF CHAIN LIST.START REGISTER LIST.                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

REGLIST  YEC;

      FALSE              4;

        FALSE.A;
        FALSE.B;
        FALSE.C;
        FALSE.E #

      UNUSEDCOPM         4;

        UNUSED.A;
        UNUSED.B;
        UNUSED.C;
        UNUSED.D #

      UNUSEDTEST         2;

        OR.UNUSEDB;
        AN.UNUSEDB #

      PROSTAT            3;           % PROCESSOR STATE COMPARISON REG.

        PROSTA-2;
        PROSTA-1;
        PROSTA-0 #

      PROSTATCEN         3;           % PROC STATE COMPARISON ENABLE REG

        PROSTASL-2;
        PROSTASL-1;
        PROSTASL-0 #

REGEND;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             END OF MAINTENANCE CHAIN REGISTER LIST.                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

XM DATA CARD CHAIN LIST (MAINTENANCE)

MODULE NAME : XMD DATE : 6/25/85

```
DATE     J012387;                %

CHAIN     YED;                   %XM DATA CARD MAINTENANCE CHAIN.
                                 % 112 BITS LONG.

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %      CLKMNT2 ARRAY.  YED[111-96]                             %
      %      ------------------------                                %
      %      CHAIN LENGTH --> 16 BITS.                               %
      %      LOCATION ------> R6F8.                                  %
      %      SDI PIN -------> P08.                                   %
      %      SDO PIN -------> L13.        PART # : 1996 0053         %
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        ERRORIG          1;        % error ignore
        OVERRUN          1;        % over run
        COUNTERFF        1;        % counter f/f
        CLKBAD           1;        % clock bad
        SKEWLOW          1;        % skew low f/f
        SKEWHI           1;        % skew hi f/f
        MODIFYSKEW       1;        % modify skew f/f
        SKEWREG          9;        % skew register

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %      STOP ARRAY.  YED[95-0]                                  %
      %      --------------------                                    %
      %      CHAIN LENGTH --> 96 BITS.                               %
      %      LOCATION ------> K4D4.                                  %
      %      SDI PIN -------> C15.                                   %
      %      SDO PIN -------> H14.        PART # : 1995 7307         %
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        AND.FF           1;        % AND STOP FF.
        OR.FF            1;        % OR STOP FF.
        PRETRIGGER       1;        % PRETRIGGER.
        MSDATAHIT        1;        % REMEMBER MS DATA HIT.
        MSADDRHIT        1;        % REMEMBER MS ADDRESS HIT.
        NOTUSED3         6;        % NOT USED
```

```
                ANMSDATHIT      1;      % STOP AND,ON REMEMBERED MSDATHIT.
                NOTUSED6        1;      % not used.
                ANMSADRHIT      1;      % STOP AND,ON REMEMBERED MSADRHIT.
                NOTUSED4       19;      % not used.
                AND.FPMARK      1;      % STOP AND ENABLE ON FPAGE MARKED.
                OR.FPMARK       1;      % STOP OR ENABLE ON FPAGE MARKED.
                NOTUSED5       12;      % not used.
                MS.DATA        16;      % MASS STORE STOP DATA VALUE
                MS.ADDR         8;      % MASS STORE STOP ADDRESS VALUE
                S/B.FALSE      10;      % 40 BIT COMPARE SELECTS -NOT USED
                MSDATCOMEN      1;      % ENABLE COMPARE OF MSTORE DATA
                MSADRCOMEN      5;      % ENABLE COMPARE OF MSTORE ADDRESS
                BPSTOPEN        1;      % BACK PLANE STOP ENABLE
                NOTUSED2        1;      %
                NOT.MSADR       1;      % STOP IF NOT EQUAL TO MSADR REG.
                OR.MSDATA       1;      % MASS STORE DATA STOP OR ENABLE.
                OR.MSADR        1;      % MASS STORE ADDR STOP OR ENABLE.
                NOTUSYED1       1;      %
                AND.MSDATA      1;      % MASS STORE DATA STOP AND ENABLE.
                AND.MSADR       1;      % MASS STORE ADDR STOP AND ENABLE.
                NOTUSED0        1;      %

    CHNEND;                             % END OF CHAIN

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %               END OF CHAIN LIST.START REGISTER LIST.                        %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    REGLIST  YED;
                                        %THERE ARE NO REGISTERS IN THIS
                                        %CHAIN.
    REGEND;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %               END OF MAINTENANCE CHAIN REGISTER LIST.                       %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

7.2 STOP LOGIC

7.2.1 STOP MODES

There are three modes of stopping clocks in the XM: the livefreeze, the deadfreeze, and the maintenance stop. The first two are XM-self stops. The last is an external stop, caused by the System Maintenance Controller (SMC), making the maintenance group clock enable term false on the following clock. The maintenance control logic, on each XM card, freezes the entire module except for the maintenance paths.

7.2.2 STOP CONDITIONS

Conditions causing an XM to stop itself, or to be stopped, are the following:

- functional, non-error conditions (detected by the XM) in which the XM must wait for an external event to occur before continuing. Such conditions always cause the livefreeze (See 6.2.1).

- fatal internal error conditions, detected by the XM, which cause the deadfreeze stop and flag the SMC. The ERROROK and ERROR bit on the chain, when set, inhibits the deadfreeze stop from taking place and prevents invoking the SMC. Internal error detection is described in 7.3.

- Selectable stop on microcode address match. The BRKPT register on the maintenance chain is compared to the next micro-instruction address and on the next clock; a flip flop sets to indicate a match. When there is a match, a signal is sent to the SMC. If the address stop enable is on, livefreeze takes place on this clock. The CREG will contain the micro-instruction of the stop address, since the clock stops prior to execution.

- External conditions, such as: errors detected within other modules or other system-related events, which result in a maintenance stop.

7.2.3 INTERNAL SOURCES TO XM MAINTENANCE STOP LOGIC

There are two STOP LOGIC arrays on the XM. One on each board.

-Inputs to stop logic on ED

1. Mass store address - 8 bit compare.
2. Mass store data - 16 least significant bits
3. Task Number bit
4. Environment number bit

-Inputs to stop logic on EC

1. Processor State - 3 bit compare.
2. EUPC - 14 bit compare.
3. Fetch Flush - bit-high test.
4. System Flush - bit-high test.
5. Normal IO Complete
6. Real Time IO Complete
7. IO Error
8. Timeout 1
9. Fault
10. Corrected Error
11. Interrupt
12. Fetch Page Marked
13. Task Timer Interrupt
14. External trigger

7.3 ERROR DETECTION

The XM detects the following fatal errors:

1) Parity error in data path

2) FBus and RBus Parity Error

3) Stack Overflow

4) Stack Underflow

5) Instruction Timeout 2

Whenever one of these conditions is detected, deadfreeze takes place and MOD_BROKEN is sent to the SMC, unless the ERROROK is set. Each of these errors, sets a bit in the control chain in addition to the deadfreeze flag.

7.3.1 PARITY ERRORS

Parity is carried throughout the data path, and to a certain extent, through sections of the control lines. An error detected by the receiver sets a F/F to indicate the fault. The output of this F/F is used to hold the state of the module for one clock until the clock chip turns clocks off.

F-BUS PARITY ERROR

A single bit error detected at the input to the FBUS, in either the control or data field, will get reported as a future op error. In addition, the XM will look at the NOTRY F/F (Section 7.6) and the field that indicates an OPLIT branch before determining when to stop the module.

If the XM is in a retryable state and an error is detected, then a deadfreeze condition is reported. If the XM is in a non-retryable state and an error is detected, it will wait until the next OPLIT branch before deadfreezing.

7.3.1 PARITY ERRORS (Continued)

RBUS PARITY ERROR

A single bit error in the RBUS data or control field will set a deadfreeze F/F.

CREG PARITY

CREG Parity error stop, freezes the XM immediately upon detection of a uWord parity error and sets CREGERR This will preserve data integrity to allow the maintenance subsystem to attempt micro-instruction retry (See Section 7.7).

ICU PARITY

Each data source to the ICU has 2 parity bits asscciated with it. The ICU has a 2 bit path which transfers the paritys. It is checked at each receiving function to validate data integrity through the ICU. Any ICU Parity error will set an ICU Parity Error flag which will cause a deadfreeze.

MASS STORE PARITY

Mass Store has ECC stored with the data which provides Single Bit Correct/Double Bit Detect (SBC/DBD) if done by the SMC. Refer to Appendix D.

7.3.2 STACK ERRORS

The microcode stack may be incorrectly used either by hardware fault or microcode error. The discernable errors are overflow and underflow, as described in the following paragraphs.

STACK OVERFLOW ERROR

Stack overflow, i.e. trying to push the fifth entry into the four deep stack, sets a bit (SOVFLO), in the stackpointer control logic, flagging the overflow condition. Stack overflow also sets deadfreeze (DFREEZ).

STACK UNDERFLOW ERROR

Trying to pop an empty stack will set a bit (SUNFLO), in the stackpointer control logic, flagging the stack underflow condition and setting deadfreeze (DFREEZ).

7.3.3 uINSTRUCTION REDUNDANT MUX COMPARISON

In order to provide necessary fanout the top 12 bits of the ucode address multiplexer must be replicated. An output of one instance of the mux is fed to a comparator on the next to verify that they are outputting the same address. Each instance has its own comparator and will be chained if more than two instances are required.

7.4 HISTORY FILE

The history file stores the uPC value of the last two calls or branches to allow back tracing of microcode from an error or stop condition.

The addresses found in this file are "address of branch" plus one. The history file pointer will point to the oldest branch address.

7.5 uWORD BREAKPOINT ADDRESS REGISTER

This 14 bit register holds a micro-instruction address that gets compared to the output of the micro-instruction mux. It is shifted via the maintenance chain and the comparison is also enabled by a bit in the maintenance chain (STOPEN). If a compare equal is detected and STOPEN is true, the clocks to the module are stopped on the following clock. At the same time, the SMC is informed that a match has been found.

7.6 INSTRUCTION RETRY

After certain fatal errors inside the processor, MP can retry the instruction by flushing FETCH to the current PC inside XM, provided that the NOTRY flag is not set.

The XM microcode sets the NOTRY flag once a critical state inside the processor or memory subsystem is altered, e.g., after a memory write or updating of the global COMS.

The PC is obtained from the current fetch page entry 4.

7.7 MICRO-INSTRUCTION RETRY

Micro-instruction retry may take place under only one condition: CREG parity error.

On a CREG parity error, the SMC is notified, via the XM_STOP_TIME wire, prior to the execution of the micro-instruction and can read the UPC to get the address at fault (actually UPC-1). With this information, the maintenance system can rewrite the CRAM with correct data and allow the XM to continue.

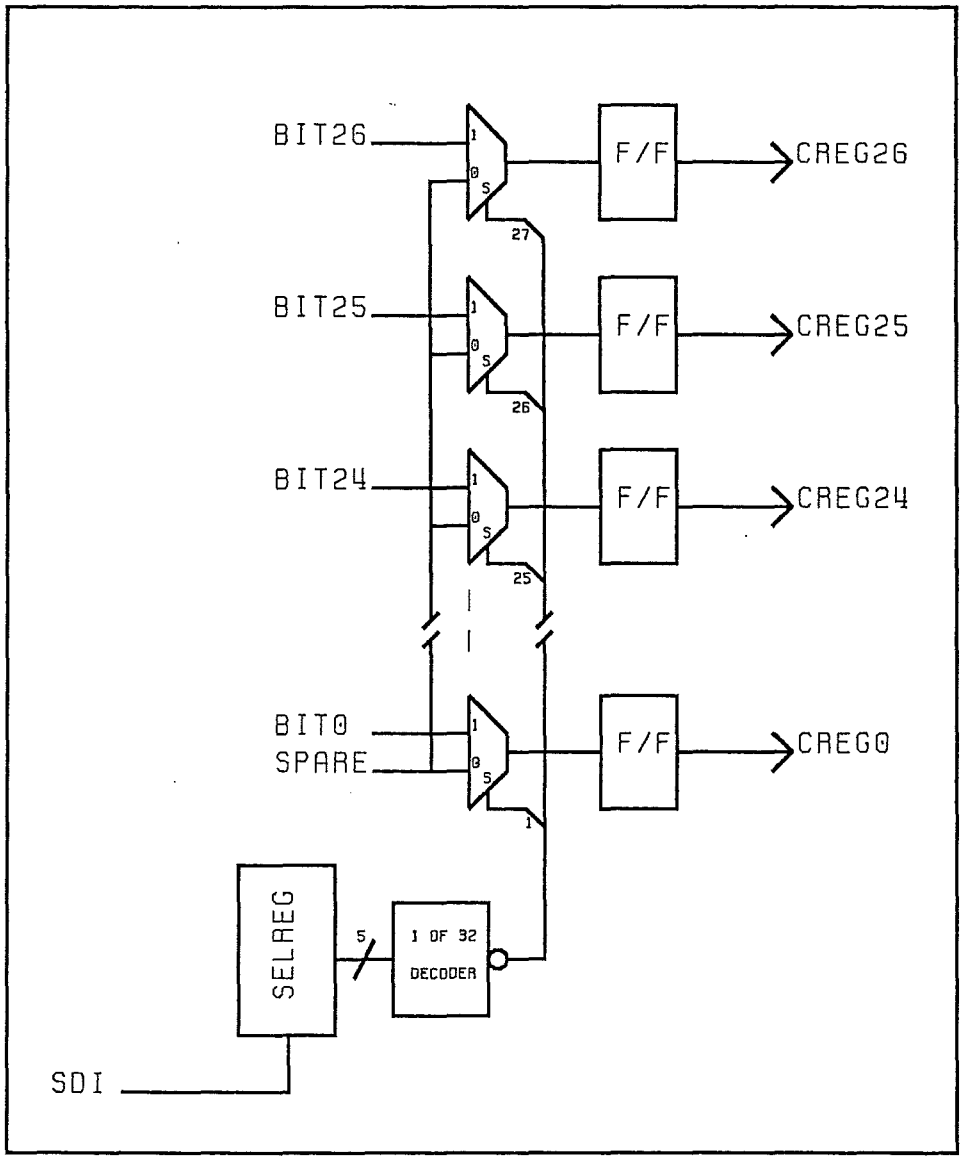
7.7.1 CONTROL STORE SPARE RAM ALGORITHM

A spare RAM exists to increase the Mean-Time between repairs due to a single RAM failure in Control Store.

Assuming the module has detected a parity error in Control Store, and this is a single bit error, then the MP would start the recovery algorithm. First, by shifting the chain out, it would read the contents of the UPC (micro program counter) to determine which microword is in error. Second, it will compare the contents of the CREG with those stored on disk to isolate the bit at fault. Once the bit position is known, the software would shuffle the bit to be replaced into a bit in the chain that loads the spare column. Now the data is ready to be transferred to the module. To prepare the module for the switch between the spare and the error bit, the MP would load the value corresponding to the CREG-bit-in-error in the SELREG. It should be pointed out that the S/W must be smart enough to map bit n to a value between 1 and 27 before selecting which slice should receive it on it's SELREG. If the other SELREG's contain a value of zero, then it's spare is disabled.

Figure 7-1 is a block diagram of the 27 2-1 multiplexers along with their SELREG/DECODER logic. The 1 of 32 selects one, if any, of the bits to be swapped with the spare column. This path completes the replacement algorithm.

FIGURE 7-1 SPARE RAM MUX TO CREG



7.8 MEASUREMENT OP REGISTER (MOPREG)

MOPREG is a 32 bit register that loads via the 32 least significant bits of the ICU SPIN mux and routes the contents to the backplane for use in software measurements. Loading of the register is enabled by an internal command (INTCMD=03).

7.9 MASS STORE ECC ERROR

On a mass store parity error, the SMC is notified prior to the use of the data, the clock having been stopped before execution of the microword, or the reloading of the mass store register. With the address of the faulty word in the mass store register, the data from mass store and the ECC bits from the mass store ECC register (ECCREG), the SMC can calculate the correct contents of that location. The data may be rewritten into mass store and the execution of the microcode continued.

8 TESTABILITY

8.1 MEMORY REQUESTOR TEST BUS

This bus is made up of control and status bits from all of the requestor units. The function of this bus is to route these control bits over to the data path via the LPCTR lines of REAM.

Requestor command 1C will put the unit into the Test Mode, where the AW bus drivers will be disabled and the control information will be visible on the LPCTR lines. Various commands will activate various control/status bits. Requestor command 1D will put the unit back into the normal mode of operation. For details on the control/status bits in the Tbus, refer to Appendix E. For details on which control/status bit is on during a particular command, refer to the functional test microcode for the memory requestor.

8.2 INTERNAL FBUS

Internal Fbus allows the function test microcode to write into the Fetch pages and set various Fetch page related flags without having anything to do with Fetch itself.

The bus is enabled via an internal command (ICMD=08) and disabled via another internal command (ICMD=09). Having enabled the Internal Fbus, the test microcode can do any of the following. See Figure 8-1.

a. Write into Fetch page 1, any page, any entry.

b. Write into Fetch page 2, any page, any entry.

c. Set Data Hit and Optimal flags selectively for any page.

d. Selectively shove each page.

The following must also be noted.

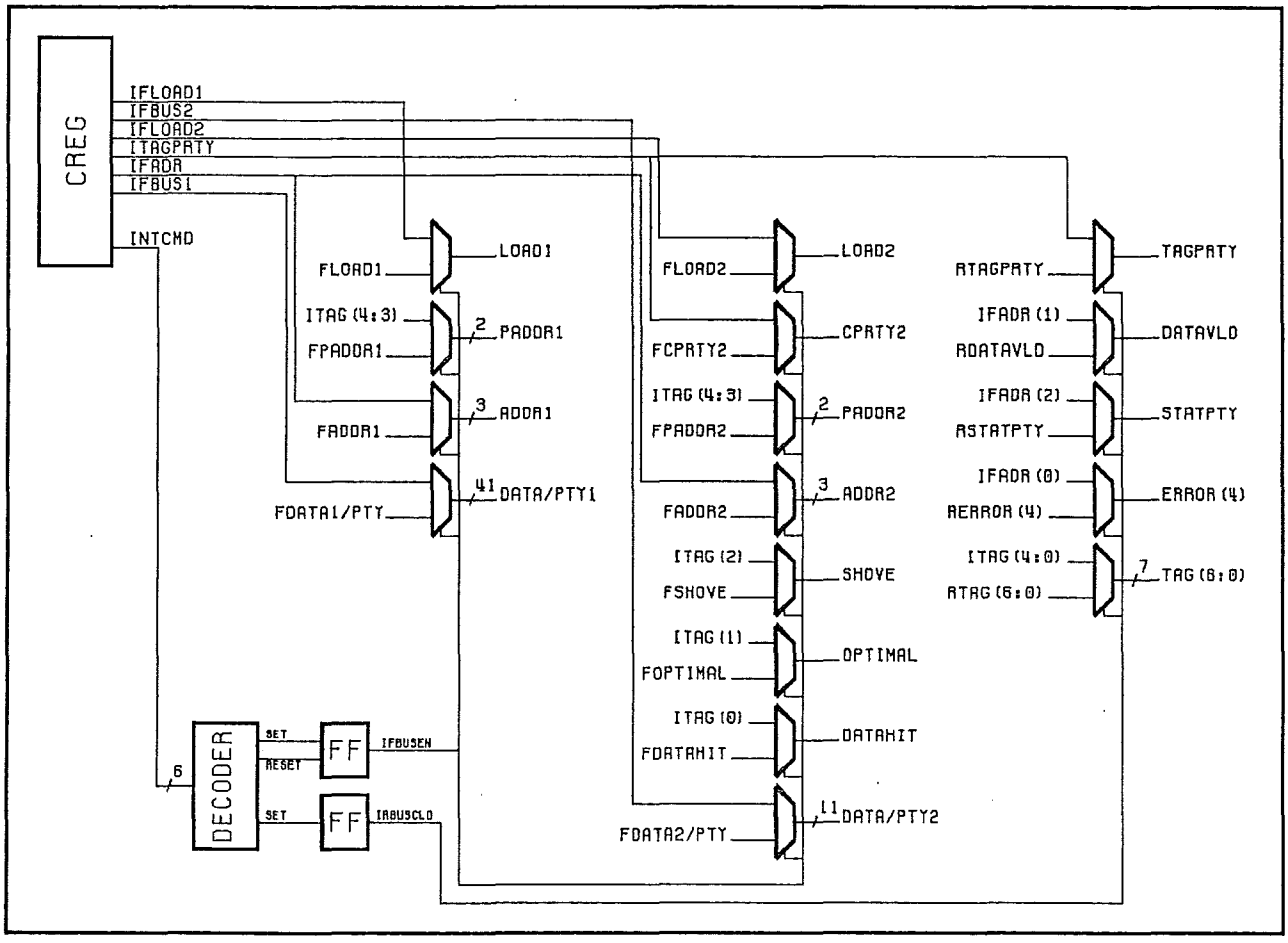
a. IFBUS1 and IFBUS2 share a common page address.

b. IFBUS1 and IFBUS2 share a common entry address.

c. IFBUS1 and IFBUS2 have separate load lines.

For a more detailed description and CREG bit assignments, refer to Appendix C.

FIGURE 8-1 INTERNAL FBUS/RBUS



8.3 INTERNAL RBUS

Internal Rbus allows the Function Test microcode to set RBUS0/OP0 valid bits, without actually writing the data, as well as cause faults and non-fatal errors.

The bus is enabled via an internal command (1A) and to stay enabled, the command must stay valid throughout the entire operation. Once the command is switched, the internal RBUS will be disabled. Refer to Appendix C. See Figure 8-1.

INTERNAL RBUSC TAG GENERATION

The reason ITAG(6:5) are not present is that ITAG(4:4)=1 will produce a Write Tag, and ITAG(4:4)=0 will produce an RBUS/OPRND queue tag based on ITAG(3:3). The following table will illustrate the tag generation.

ITAG(4:0)	ACTUAL INTERNAL TAG GENERATED	
10XXX	1010XXX	
00XXX	1100XXX	;RBUS0 TAG
01XXX	1101XXX	;OP0 TAG

Where XXX has the following meanings:

ITAG(4:3)	INTERPRETATION OF XXX (X2, X1, X0
10	X2, X1 = XM page number, X0=0
00	X2, X1, X0 = RBUS queue entry number
01	X2, X1 = XM page number X0 = 0 --> Operand A X0 = 1 --> Operand B

8.3 INTERNAL RBUS (Continued)

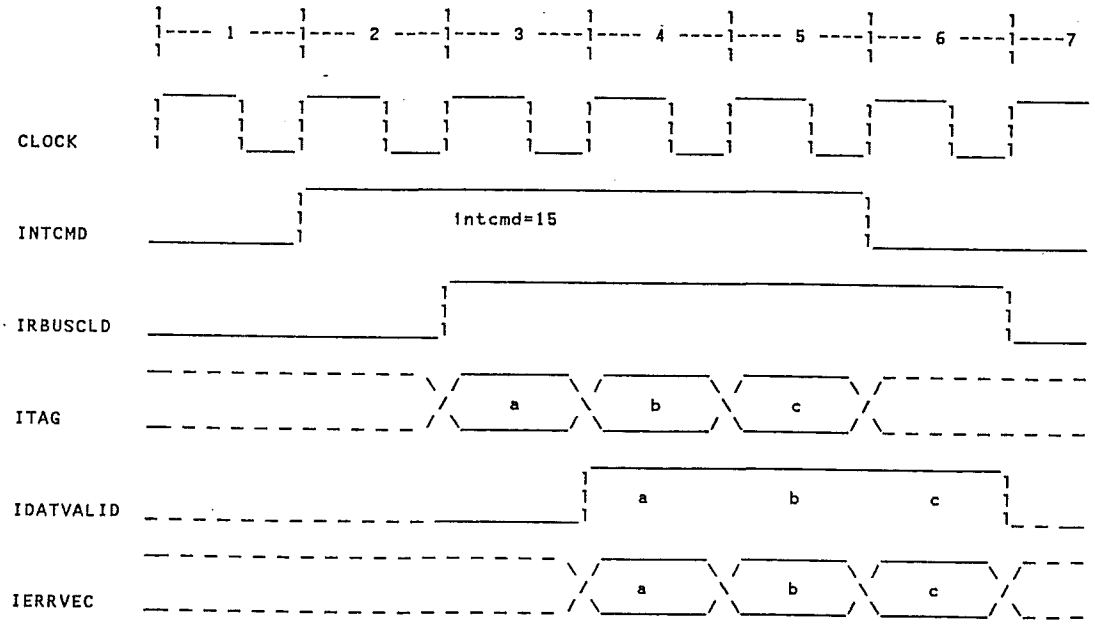
INTERNAL RBUSC ERROR VECTOR GENERATION

The setup for the Error Vector Generation does not allow for all errors to be generated. Only 1 nonfatal error can be generated, and 1 fault can be generated. The following table will illustrate the Error Vector Generation.

IERRVEC(4:4)	ERROR VECTOR GENERATED
0	C0CC1 (Non fatal error)
1	100C1 (Fatal error = fault)

Note: IDATVALID must be "ON" the clock after ITAG has been sent, and at the same clock IERRVEC is sent in order to accept the information. Refer to the following timing diagram for details:

FIGURE 8-2 INTERNAL BUS TIMING



APPENDIX A - PTEST FUNCTIONS

SUMMARY OF THE PTEST FUNCTIONS

Convention:

PTIN:	PTEST data input
D9,D8,D7,D6,D5,D4,D3,D2,D1,D0:	Digits in PTIN
PTREG	PTEST register It is either the current value or the new value of the PTEST register (depends on context)
B3,B2,B1,B0:	Bits in PTREG
[m,n]:	Bits, start from mth bit for n bits (m, m-1, m-1,..., m-n+1).

PTEST(0):

```
B3 = 0   If PTIN.[31:4] <> 0   (Error report from FETCH )
B2 = 0   If undigit in PTIN
B1 = 0   If PTIN.[0:1]=1       (Odd address)
B0 = 0   If Digit 6-0 undigit  (Address undigit)
```

PTEST(1): Used for Environment Table Entry and Memory Area Table Entry check.

```
B3,B2 = Don't care
B1,B0 = 00   If 0 <= D9 <= 9
        01   If D9 = 4"D"
        10   If D9 = 4"C"
        11   Else   (=4"A",4"B",4"E",or 4"F")
```

APPENDIX A - PTEST FUNCTIONS (Continued)

PTEST(2): Leading zeros detection
PTREG = # of leading zeros in PTIN (10 coded as 1010B)

```
      = 0    if PTIN = D XXXXXXXXX
        1    if PTIN = 0 D XXXXXXXX
        2    if PTIN = 00 D XXXXXXX
        3    if PTIN = 000 D XXXXXX
        4    if PTIN = 0000 D XXXXX
        5    if PTIN = 00000 D XXXX
        6    if PTIN = 000000 D XXX
        7    if PTIN = 0000000 D XX
        8    if PTIN = 00000000 D X
        9    if PTIN = 000000000 D
        4"A" if PTIN = 0000000000
```

where
D: non-zero digit
X: don't care

PTEST(3): Undefined
PTEST(4): Undefined
PTEST(5): Undefined

PTEST(6): Zero detection
B3 = 1 If D9="D"
B2 = 1 If D8, D6, D4, D2, and D0 are all zeros
B1 = 1 If Digit 8-0 all zero
B0 = 1 If PTIN=0

PTEST(7): Undefined

PTEST(8): For Integer OP's
B3 = 1 If D9<>0 or D8<>0 or D7<>0
B2 = 1 If D9<>0 or D8<>0
B1 = 1 If D9=0
B0 = 1 If D6="D"

APPENDIX A - PTEST FUNCTIONS (Continued)

```
PTEST(9): Siginificant digit detection
          Test for LOAD MAT routine
   B3,B2 = 00  If 1 significant digit  in Digit 9-2  (0000000D XX)
           01  If 2 significant digits in Digit 9-2  (000000DX XX)
           10  If 3 significant digits in Digit 9-2  (00000DXX XX)
           11  If 4 significant digits in Digit 9-2  (0000DXXX XX)
           00  Else (strange??)
     where D: non-zero digit
           X: Don't care

   B1 = 1  If D6=4"0"
   B0 = 1  If D8D7=4"DD"


PTEST(A): Encoded pass function
   PTREG = D0   If PTIN<"A"
         = "A"  If PTIN=4"10"
         = 0    Else


PTEST(B): Pass Digit 2
   PTREG = D2


PTEST(C): Trailing zero detection
   PTREG = 10 - number of trailing zeros in PTIN
           (10 coded as 1010B)
         = 0    if PTIN = 0000000000
           1    if PTIN = D 000000000
           2    if PTIN = X D 00000000
           3    if PTIN = XX D 0000000
           4    if PTIN = XXX D 000000
           5    if PTIN = XXXX D 00000
           6    if PTIN = XXXXX D 0000
           7    if PTIN = XXXXXX D 000
           8    if PTIN = XXXXXXX D 00
           9    if PTIN = XXXXXXXX D 0
           4"A" if PTIN = XXXXXXXXX D
       where
           D: non-zero digit
           X: don't care
```

APPENDIX A - PTEST FUNCTIONS (Continued)

PTEST(D): Undigit check
Sign detection
B3 = 0 If D9 = 4"D"
B2 = If D9 = 4"D" and D8-D0 <> 0
B1 = 0 If D9-D2 has undigit
B0 = 0 If D8-D2 has undigit

PTEST(E): Pass D0
PTREG = D0

PTEST(F): Leading zeros in UA Zone data
Check UA numeric digits only (D8,D6,D4,D2,D0)
PTREG = number of leading zeros in numeric digits

= 0	if PTIN = X D XXXXXXXX
8	if PTIN = X 0 X D XXXXXX
6	if PTIN = X 0 X 0 X D XXXX
4	if PTIN = X 0 X 0 X 0 X D XX
2	if PTIN = X 0 X 0 X 0 X 0 X D
0	if PTIN = X 0 X 0 X 0 X 0 X 0

where D: non-zero digit
X: don't care

PTEST(10): Compute |AF-BF|
Input: PTIN = 0000 0PAFBF
Output: PTREG = |AF-BF|
Notes: 1. Input data in BCD format
2. Output data in BCD format
3. If AF indicates a literal, the actual length of literal is used. (e.g. A3 -> 3, A9 -> 2, B2 -> 4)
4. To simplify hardware implementation, it can be assumed that the final lengths of A & B are less or equal to 4"10"

APPENDIX A - PTEST FUNCTIONS (Continued)

PTEST(11): Undefined
PTEST(12): Undefined
PTEST(13): Undefined
PTEST(14): Undefined
PTEST(15): Undefined
PTEST(16): Soft PTEST FUNCTION1
PTEST(17): Soft PTEST FUNCTION2

Followings are PTEST2 functions which provide operations across 40 bits of PTIN and the current value of PTREG

PTEST(18): Compare
B3 = 1 If any zone digit is zero (e.g. XX0XXXXXXX)
B2 = 0 If any byte digit is zero (e.g. XXXX00XXXX)
B1 = 1 If D0=PTREG
B0 = 1 If D0>PTREG

PTEST(19): Compare leading zeros
B3 = 0 If PTIN <> 0
B2 = 0 If number of leading zeros in PTIN < PTREG
B1 = 0 If number of leading 9's in PTIN < PTREG
B0 = 0 If number of trailing zeros in PTIN < PTREG

PTEST(1A): Undefined
PTEST(1B): Undefined
PTEST(1C): Undefined
PTEST(1D): Undefined
PTEST(1E): Undefined
PTEST(1F): Undefined

APPENDIX B - MCACM ERROR VECTOR DEFINITION

MCACM ERROR VECTOR DEFINITION TABLE FOR V5ØØ AS INTERPRETED BY THE 'XM' MODULE

	F/C	XMW	XMR	ERROR	FLT	NTY	MCERR	INTERPRETATION
1-	Ø	Ø	Ø	ØØ	Ø	Ø	ØØ	NO ERRORS OR FAULTS REPORTED.
2-	Ø	Ø	1	Ø1-ØF	Ø	Ø	41-4F	NON FATAL ERROR CAUSED BY XM READ.(SETS TEST CONDITION)
3-	Ø	1	Ø	Ø1-ØF	Ø	Ø	81-8F	NON FATAL ERROR CAUSED BY XM WRITE.(SETS TEST CONDITION)
4-	1	Ø	Ø	Ø1-ØF	Ø	Ø	21-2F	NON FATAL ERROR CAUSED BY FETCH.(SETS TEST CONDITION)
5-	Ø	Ø	1	1Ø	Ø	Ø	5Ø	MULTIPLE BIT MEMORY ERROR.WILL CAUSE DEAD FREEZE.
6-	Ø	1	Ø	1Ø	Ø	Ø	9Ø	MULTIPLE BIT MEMORY ERROR.WILL CAUSE DEAD FREEZE.
7-	1	Ø	Ø	1Ø	Ø	Ø	3Ø	MULT. BIT MEM ERROR CAUSED BY FETCH.LIVE FREEZE IMMEDIATELY.
8-	1	Ø	Ø	1Ø	Ø	1	3Ø	MULT. BIT MEM ERROR CAUSED BY FETCH.LIVE FREEZE AT OPLIT BRANCH.
9-	1	Ø	1	1Ø	Ø	Ø	7Ø	MULT. BIT MEM ERROR CAUSED BY 'OF' PREREAD.LIVE FREEZE IMMEDIATELY.
1Ø-	1	Ø	1	1Ø	Ø	1	7Ø	MULT. BIT MEM ERROR CAUSED BY 'OF' PREREAD.LIVE FREEZE AT OPLIT BRANCH.
11-	Ø	Ø	1	13-1F	Ø	Ø	53-5F	FATAL ERROR.WILL CAUSE DEAD FREEZE.
12-	Ø	1	Ø	13-1F	Ø	Ø	93-9F	FATAL ERROR.WILL CAUSE DEAD FREEZE.
13-	1	Ø	Ø	13-1F	Ø	Ø	33-3F	FATAL ERROR CAUSED BY FETCH.LIVE FREEZE IMMEDIATELY.
14-	1	Ø	Ø	13-1F	Ø	1	33-3F	FATAL ERROR CAUSED BY FETCH.LIVE FREEZE AT OPLIT BRANCH.
15-	1	Ø	1	13-1F	Ø	Ø	73-7F	FATAL ERROR CAUSED BY 'OF' PREREAD.LIVE FREEZE IMMEDIATELY.
16-	1	Ø	1	13-1F	Ø	1	73-7F	FATAL ERROR CAUSED BY 'OF' PREREAD.LIVE FREEZE AT OPLIT BRANCH.
17-	Ø	Ø	1	11-12	Ø	Ø	51-52	FAULT CAUSED BY XM READ:INTERRUPT AT OPLIT BRANCH.
18-	Ø	1	Ø	11-12	Ø	Ø	91-92	FAULT CAUSED BY XM WRITE.INTERRUPT AT OPLIT BRANCH.

APPENDIX B - MCACM ERROR VECTOR DEFINITION (Continued)

19-	1	Ø	Ø	11-12	Ø	Ø	31-32	FAULT CAUSED BY FETCH.NO ACTION.(WILL BE HANDLED BY 'OF')
2Ø-	1	Ø	1	11-12	Ø	Ø	71-72	FAULT REPORTED FOR A FUTURE PAGE.INVALIDATED BY A FLUSH.
21-	1	Ø	1	11-12	1	Ø	71-72	FAULT REPORTED FOR PAGE Ø.INTERRUPT AFTER EXECUTING PAGE Ø.
22-	1	Ø	1	11-12	2	Ø	71-72	FAULT REPORTED FOR PAGE 1.INTERRUPT AFTER EXECUTING PAGE 1.
23-	1	Ø	1	11-12	4	Ø	71-72	FAULT REPORTED FOR PAGE 2.INTERRUPT AFTER EXECUTING PAGE 2.
24-	1	Ø	1	11-12	8	Ø	71-72	FAULT REPORTED FOR PAGE 3.INTERRUPT AFTER EXECUTING PAGE 3.

EXPLANATION OF TERMS:

1- F/C = FUTURE/CURRENT BIT INDICATOR --> (NAME ON CHAIN = FUTCURER)
2- XMW = INDICATES TAG RETURNED IS FOR AN XM WRITE --> (NAME ON CHAIN = XMWRITE)
3- XMR = " " " " " " READ --> (NAME ON CHAIN = XMREAD)
4- ERROR = MCACM ERROR VECTOR --> (NAME ON CHAIN = ERRORVEC)
5- FLT = FUTURE FAULT INDICATORS --> (NAME ON CHAIN = FAULT)
6- NTY = NOTRY FLAG,IF ON ,XM IS NOT RETRYABLE --> (NAME ON CHAIN = NOTRYA,NOTRYB 2 COPIES)
7- MCERR = COMBINATION OF THE FIRST FOUR FIELDS --> (NAME ON CHAIN = MCERROR)

XMW	XMR	F/C	ERROR	=	MCERROR

8- ALL LIVE FREEZES MENTIONED ARE REFFERING TO FUTURE ERROR BIT IN THE 'LIVEFRZ' REGISTER.

APPENDIX C - INTERNAL FBUS AND RBUS CONTROL BIT DEFINITIONS

INTFBUS DEFINITION:

IFBUS1 FIELD DESCRIPTION	ARRAY RESIDENCE	CREG FIELD (BIT #)	CREG FIELD DESCRIPTION
IDATA(00:00)	FPQ1 SLICE 0	CREGB(046:046)	QCCNCTFNC(2:2)
IDATA(02:01)	FPQ1 SLICE 0	CREGB(048:047)	ECCNCTFNC(4:3)
IDATA(05:03)	FPQ1 SLICE 0	CREGB(051:049)	ECPMBFUNC(2:0)
IDATA(06:06)	FPQ1 SLICE 1	CREGB(052:052)	ECPMBFUNC(3:3)
IDATA(07:07)	FPQ1 SLICE 1	CREGB(053:053)	ECPMAFUNC(0:0)
IDATA(10:08)	FPQ1 SLICE 1	CREGB(056:054)	QCPMAFUNC(3:1)
IDATA(11:11)	FPQ1 SLICE 1	CREGB(085:085)	QCLITADR(0:0)
IDATA(12:12)	FPQ1 SLICE 2	CREGB(086:086)	QCLITADR(1:1)
IDATA(17:13)	FPQ1 SLICE 2	CREGB(091:087)	QCLITADR(6:2)
IDATA(18:18)	FPQ1 SLICE 3	CREGB(092:092)	QCLITADR(7:7)
IDATA(19:19)	FPQ1 SLICE 3	CREGB(079:079)	QCMSADSEL(1:1)
IDATA(20:20)	FPQ1 SLICE 3	CREGB(099:099)	QCLUFUNC(0:0)
IDATA(21:21)	FPQ1 SLICE 3	CREGB(100:100)	QCLUFUNC(1:1)
IDATA(22:22)	FPQ1 SLICE 4	CREGB(101:101)	ECMSDCMPRH
IDATA(27:23)	FPQ1 SLICE 4	CREGB(107:103)	ECPTFUNC(4:0)
IDATA(31:28)	FPQ1 SLICE 5	CREGB(114:111)	QCSPCADR(3:0)
IDATA(33:32)	FPQ1 SLICE 5	CREGB(120:119)	QCSPBADR(1:0)
IDATA(35:34)	FPQ1 SLICE 6	CREGB(122:121)	QCSPBADR(3:2)
IDATA(39:36)	FPQ1 SLICE 6	CREGB(126:123)	QCSPAADR(3:0)
IFLOAD1	FRQCTL	CREGB(058:058)	PMBSEL(1:1)
IFPARITY1	FRQCTL	CREGB(084:084)	QCLITSELH

APPENDIX C - INTERNAL FBUS AND RBUS CONTROL BIT DEFINITIONS
(Continued)

IFBUS2 FIELD DESCRIPTION	ARRAY RESIDENCE	CREG FIELD (BIT #)	CREG FIELD DESCRIPTION
IDATA(0:0)	FPQ2 SLICE 0	CREGB(029:029)	QCAUFUNC(0:0)
IDATA(1:1)	FPQ2 SLICE 0	CREGB(078:078)	QCMSADSEL(0:0)
IDATA(2:2)	FPQ2 SLICE 0	CREGB(117:117)	QCSPINSELH(2:2)
IDATA(3:3)	FPQ2 SLICE 0	CREGB(044:044)	QCCNCTFNC(0:0)
IDATA(4:4)	FPQ2 SLICE 0	CREGB(062:062)	QCRQLSELH(0:0)
IDATA(6:5)	FPQ2 SLICE 1	CREGB(064:063)	QCRQLSELH(2:1)
IDATA(8:7)	FPQ2 SLICE 1	CREGB(072:071)	QCRQSPADR(1:0)
IDATA(9:9)	FPQ2 SLICE 1	CREGB(098:098)	QCLUASELH(1:1)
IFLOAD2	FRQCTL	CREGB(057:057)	QCPMBSELH(0:0)
ISHOVE	FRQCTL	CREGB(108:108)	QCPTSELH(0:0)
IDATAHIT	FRQCTL	CREGB(034:034)	QCAUBSELH(1:1)
IOPTIMAL	FRQCTL	CREGB(035:035)	QCAUBSELH(2:2)
IFCPRTY2	FRQCTL	CREGB(033:033)	QCAUBSELH(0:0)
IFPARITY2	FRQ2 SLICE 0/1	CREGB(045:045)	QCCNCTFNC(1:1)
IFADR (IFBUS1/IFBUS2)	FPQCTL	CREGB(032:030)	QCAUFUNC(3:1)
IFPADR (IFBUS1/IFBUS2)	FPQCTL	CREGB(110:109)	QCPTSELH(2:1)

APPENDIX C - INTERNAL FBUS AND RBUS CONTROL BIT DEFINITIONS (Continued)

INTRBUSC DEFINITION:

IRBUSC FIELD DESCRIPTION	ARRAY RESIDENCE	CREG FIELD (BIT #)	CREG FIELD DESCRIPTION
ITAG(4:2)	FRQCTL	CREGB(110:108)	QCPTSELH(2:0)
ITAG(1:0)	FRQCTL	CREGB(035:034)	QCAUBSELH(2:1)
ITAGPRTY	FRQCTL	CREGB(033:033)	QCAUBSELH(0:0)
IDATVALID	FRQCTL	CREGB(031:031)	QCAUFUNC(2:2)
IERRVEC(4:4)	FRQCTL	CREGB(030:030)	QCAUFUNC(1:1)
ICTLPRTY	FRQCTL	CREGB(032:032)	QCAUFUNC(3:3)

NOTE: Internal FBUS is enabled by INTCMD=08.
Internal RBUSC is enabled by INTCMD=1A.
IFCPRTY2 is parity over IOPTIMAL, IDATAHIT, ISHOVE, IFPADR.
ICTLPRTY is parity over IERRVEC(4:4), IDATVALID.
ITAGPRTY is parity over ITAG(0), ITAG(1), ITAG(2), ITAG(3).
The usage of IRBUSC and IFBUS(1/2) must be mutually exclusive for correct operation.

APPENDIX D - MASS STORE ECC/SYNDROME

SYNDROME								BIT IN ERROR
H6	H5	H4	H3	H2	H1	H0	(HEX)	
0	X	X	X	X	X	1	--	Double bit in error
0	X	X	X	X	1	X	--	Double bit in error
0	X	X	X	1	X	X	--	Double bit in error
0	X	X	1	X	X	X	--	Double bit in error
0	X	1	X	X	X	X	--	Double bit in error
0	1	X	X	X	X	X	--	Double bit in error
1	0	0	0	0	0	1	41	H0 is in error
1	0	0	0	0	1	0	42	H1 is in error
1	0	0	0	0	1	1	43	D0 is in error
1	0	0	0	1	0	0	44	H2 is in error
1	0	0	0	1	0	1	45	D1 is in error
1	0	0	0	1	1	0	46	D2 is in error
1	0	0	0	1	1	1	47	D3 is in error
1	0	0	1	0	0	0	48	H3 is in error
1	0	0	1	0	0	1	49	D4 is in error
1	0	0	1	0	1	0	4A	D5 is in error
*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*
1	0	0	1	1	1	1	4F	D10 is in error
1	0	1	0	0	0	0	50	H4 is in error
1	0	1	0	0	0	1	51	D11 is in error
1	0	1	0	0	1	0	52	D12 is in error
*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*
1	0	1	1	1	1	1	5F	D25 is in error
1	1	0	0	0	0	0	60	H5 is in error
1	1	0	0	0	0	1	61	D26 is in error
1	1	0	0	0	1	0	62	D27 is in error
*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*
1	1	0	1	1	1	0	6E	D39 is in error
*	*	*	*	*	*	*	XX	XX * % these codes 6F-7F
*	*	*	*	*	*	*	XX	XX * % will never happen
1	0	0	0	0	0	0	40	H6 is in error
0	0	0	0	0	0	0	00	No bit in error

Note: H6-H0 are the Syndrome bits.
D39-D0 are the Data bits.

* Denotes continuation of the seouence.

APPENDIX D - MASS STORE ECC/SYNDROME (Continued)

XM MASS STORE ECC CALCULATION

		H5	H4	H3	H2	H1	H0
	A00						*
	A01					*	
-	D00					*	*
	A02				*		
-	D01				*		*
-	D02				*	*	
-	D03				*	*	*
	A03			*			
-	D04			*			*
-	D05			*		*	
-	D06			*		*	*
-	D07			*	*		
-	D08			*	*		*
-	D09			*	*	*	
-	D10			*	*	*	*
	A04		*				
-	D11		*				*
-	D12		*			*	
-	D13		*			*	*
-	D14		*		*		
-	D15		*		*		*
-	D16		*		*	*	
-	D17		*		*	*	*
-	D18		*	*			
-	D19		*	*			*
-	D20		*	*		*	
-	D21		*	*		*	*
-	D22		*	*	*		
-	D23		*	*	*		*
-	D24		*	*	*	*	
-	D25		*	*	*	*	*
	A05	*					
-	D26	*					*
-	D27	*				*	
-	D28	*				*	*
-	D29	*			*		

APPENDIX D - MASS STORE ECC/SYNDROME (Continued)

XM MASS STORE ECC CALCULATION

	H5	H4	H3	H2	H1	H0
- D30	---	**	----------	**	----------	**
- D31	---	**	----------	**	**	----------
- D32	---	**	----------	**	**	**
- D33	---	**	**	----------	----------	----------
- D34	---	**	**	----------	----------	**
- D35	---	**	**	----------	**	----------
- D36	---	**	**	----------	**	**
- D37	---	**	**	**	----------	----------
- D38	---	**	**	**	----------	**
- D39	---	**	**	**	**	----------

NOTES: -During ECC generation, A5-A0 are equal to 0.
-During ECC checking, A5-A0 are equal to H5-H0.
-In addition to H5-H0, there is an H6 bit, that is the parity over H5-H0 and D39-D00 (46 bit parity).
-Dashes in the leftmost column identify the data bits.

APPENDIX E - MEMORY REQUESTOR TEST BUS CONFIGURATION

The following table will summarize the configuration of the Test Bus (TBUS) as it applies to both REAM slices.

ACTUAL TBUS BIT NUMBER	SIGNAL NAME	REAM SLICE (LS/MS)	SIGNAL SOURCE
0	QEMPTY	LS	RELM
1	RBOPQSEL	LS	RECTAG
2	LDICMD	LS	RELM
3	RSTICMD	LS	RELM
4	REQ_BUSY	LS	RECTAG
5	LENCNTDN	LS	RECTAG
6	LPCSEL	LS	RECTAG
7	L_WRITE	LS	RECTAG
8	READREQ	LS	RECTAG
9	LDLENBUF	LS	RECTAG
10	AUFNC	LS	RECTAG
11	WRITE	LS	RECTAG
12	LDLSDADR	LS	RELM
13	NLSDDN_	LS	RELM
14	NLSDUP_	LS	RELM
15	AWRQST	LS	RECTAG
19-16	ADLSD(3:0)	LS	RELM
21-20	REGSL(1:0)	MS	RECTAG
22	DAT_PRTY	MS	REWAU
23	EN_REWAU	MS	REWAU
24	SYSFLUSH	MS	RECTAG
25	CMDPRTY	MS	RECTAG
26	SELINLEN	MS	RECTAG
27	TESTMODE	MS	REWAU
28	LDLENREG	MS	RECTAG
29	NOCMD	MS	RECTAG
30	LDADREG	MS	RECTAG
31	SELINADR	MS	RECTAG
32	0 (CONSTANT)	MS	------
34-33	3 (CONSTANT)	MS	------
35	AWRQST	MS	RECTAG
38-36	TGOUT(2:0) (TAG COUNTER)	MS	RECTAG
39	OPCOMPLT	MS	RECTAG

Note: Signal name sources can be found throughout the Memory Requestor functional specifications. Refer for explanation.

APPENDIX F - FETCH/SYSTEM FLUSH DETAILS

The following is a list of registers that get initialized to the indicated values during a FETCH/SYSTEM flush.

FETCH flush will initialize the following registers:

XM PAGE POINTER	(ECXMFPAGE)	initializes to 3
OPQA VALID FLAGS	(ECOPQAVALID)	initializes to 0
OPQB VALID FLAGS	(ECOPQBVALID)	initializes to 0
DATA HIT FLAGS	(ECDATAHIT)	initializes to 0
OPTIMAL FLAGS	(ECOPTIMAL)	initializes to 0
PAGE VALID FLAGS	(ECFPVALID)	initializes to 0
FUTURE FAULT INDICATORS	(ECFAULT)	initializes to 0

SYSTEM flush initializes the following registers in addition to the ones specified above under FETCH flush.

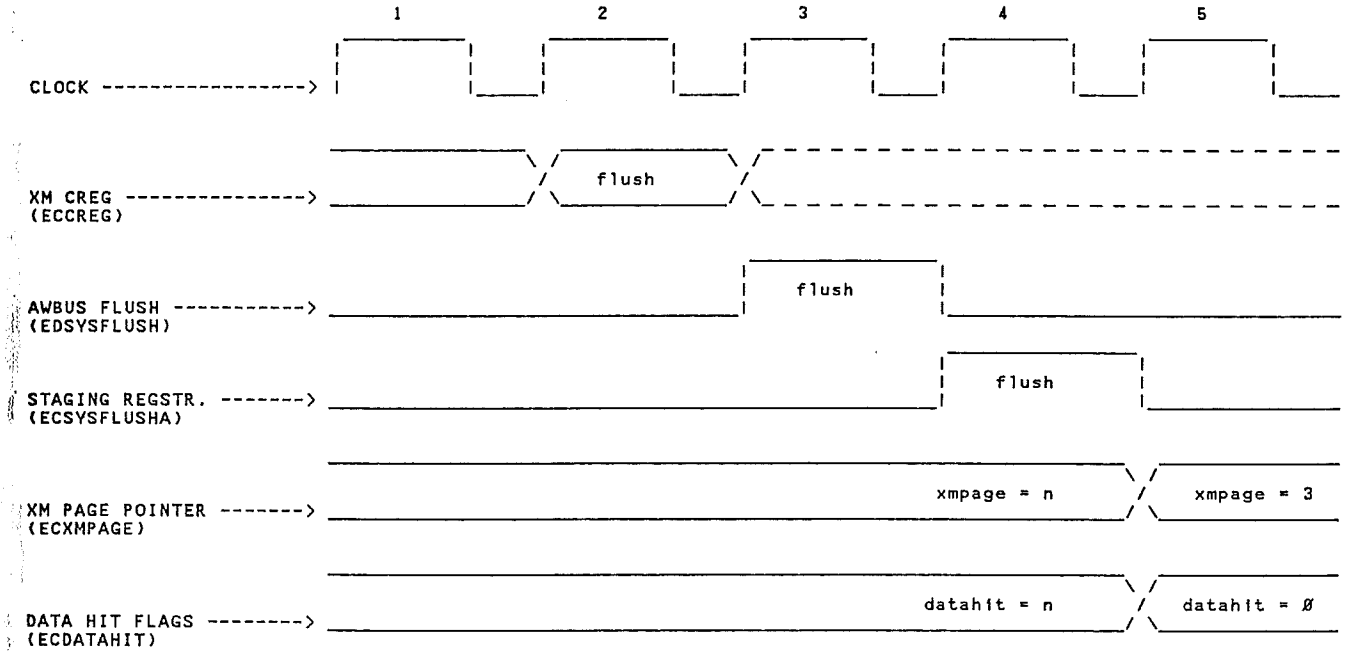
MCACM ERROR VECTOR	(ECERRVECTOR)	initializes to 0
FAULT OVERRIDE FLAG	(EDFAULTOVR)	initializes to 0

NOTES: RBUSQ read pointer (ECRBQADR) and RBUSQ VALID FLAGS (ECRBUSQVALID) are not initialized during any of the flushes. It is the responsibility of the microcoder to assure that there are no more outstanding reads in the queue.

FAULT OVERRIDE FLAG (EDFAULTOVR) is the only flag on the XM Data Card that gets initialized during a System Flush, all the others reside on the XM Control Card.

A timing diagram for SYSTEM flush follows.

APPENDIX F - FETCH/SYSTEM FLUSH DETAILS (Continued)



Timing diagram for FETCH flush is identical.

APPENDIX G - FREEZE CONTROL DIAGNOSTIC MODULE

FEATURES:

- Allows the disabling of the following errors reported to FRZCTL4:

  1- FPQ2ERR1, FPQ2ERR0.

  2- INTPERR1, INTPERR0.

  3- All data path related errors.

  4- FRQCTLER (by setting FRQERRDIS).

  5- All dead freeze conditions (by setting DFRZDIS).

- Enables the monitoring of all the error conditions feeding FRZCTL4.
- Enables the monitoring of all live freeze signals.
- Allows the disabling of all live freeze conditions.
- Allows the disabling of all dead freeze conditions.
- Allows monitoring of some internal FRZCTL4 states.
- Allows the disabling of the SPWE regardless of any live/dead freeze.
- Allows forcing the XM module into a dead freeze state for a couple of clocks and then waking it up.
- Allows forcing the XM module into a live freeze state for a couple of clocks and then waking it up.
- Helps isolate the source of the errors arriving in FRZCTL4. (whether XMC or XMD)

APPENDIX G - FREEZE CONTROL DIAGNOSTIC MODULE (Continued)

OPERATION:

The diagnostic module accepts commands via a 'TEST CODE'. This code will eventually provide the microcode with a 'FREEZE CODE'.

The 'FREEZE CODE' will relay the status of the requested signal or will transmit the signals themselves to the data path where microcode will be able to branch on.

The following is a set of tables that assist the coder to write the necessary code to test the XM freeze logic:

TO LOAD A 'TEST CODE':

1- INTCMD = 0D
2- RQCMD = 00 | load enable
3- RQSPADR-1 = 1

4- RQSPADR-0 = TEST CODE BIT 4
5- PMBFUNC-3 = TEST CODE BIT 3
6- PMBFUNC-2 = TEST CODE BIT 2
7- PMBFUNC-1 = TEST CODE BIT 1 | TEST CODE
8- PMBFUNC-0 = TEST CODE BIT 0

NOTE: -No other combination of load enable will load the TEST CODE.
-The desired TEST CODE must be present along with a load enable simultaneously.

TO RESET A 'TEST CODE':

1- INTCMD = 0E | clear 'TEST CODE'

APPENDIX G - FREEZE CONTROL DIAGNOSTIC MODULE (Continued)

TABLE OF 'TEST CODES' AND THEIR FUNCTIONS:

	TEST CODE(HEX)	FUNCTION	FREEZE CODE(binary)
1-	00	NO OPERATION	0000
2-	01	MONITOR ANY SEQ1 ERROR	1010(OFF)/1101(ON)
3-	02	MONITOR ANY SEQ2 ERROR	1010(OFF)/1101(ON)
4-	03	MONITOR ANY FPQ2 ERROR	1010(OFF)/1101(ON)
5-	04	MONITOR ANY INTRPT ERROR	1010(OFF)/1101(ON)
6-	05	MONITOR TIMER ERROR	1010(OFF)/1101(ON)
7-	06	MONITOR FRQCTL ERROR	1010(OFF)/1101(ON)
8-	07	MONITOR YECERROR	1010(OFF)/1101(ON)
9-	08	MONITOR ANY PMUX ERROR	1010(OFF)/1101(ON)
10-	09	MONITOR ANY REWAU ERROR	1010(OFF)/1101(ON)
11-	0A	MONITOR ANY REAM ERROR	1010(OFF)/1101(ON)
12-	0B	MONITOR RECTAG ERROR	1010(OFF)/1101(ON)

APPENDIX G - FREEZE CONTROL DIAGNOSTIC MODULE (Continued)

TABLE OF 'TEST CODES' AND THEIR FUNCTIONS:

	TEST CODE(HEX)	FUNCTION	FREEZE CODE(binary)
13-	0C	MONITOR RELM ERROR	1010(OFF)/1101(ON)
14-	0D	MONITOR FETCH ERROR	1010(OFF)/1101(ON)
15-	0E	MONITOR ANY DPATH ERROR	1010(OFF)/1101(ON)
16-	0F	MONITOR CREGEROR	1010(OFF)/1101(ON)
17-	10	MONITOR EPROHALT	1010(OFF)/1101(ON)
18-	11	MONITOR FUTURERR	1010(OFF)/1101(ON)
19-	12	MONITOR ENOFPAGE	1010(OFF)/1101(ON)
20-	13	MONITOR ENORDATA	1010(OFF)/1101(ON)
21-	14	MONITOR AWISBUSY	1010(OFF)/1101(ON)
22-	15	MONITOR OK2STOP	1010(OFF)/1101(ON)
23-	16	REPORT STATUS OF: OK2STOP/NOFPAGE/NORDATA/AWBUSY	
24-	17	MONITOR ETIMEOUT	1010(OFF)/1101(ON)

APPENDIX G - FREEZE CONTROL DIAGNOSTIC MODULE (Continued)

TABLE OF 'TEST CODES' AND THEIR FUNCTIONS:

	TEST CODE(HEX)	FUNCTION	FREEZE CODE(binary)
25-	18	FORCE A 2 CLOCK DEAD FREEZE	----
26-	19	FORCE A 2 CLOCK LIVE FREEZE	----
27-	1A	DISABLE FPQ2 ERROR REPORTING	----
28-	1B	DISABLE ALL DATA PATH ERRORS	----
29-	1C	DISABLE INTRPT ERROR REPORTING	----
30-	1D	REPORT STATUS OF: FUTERR/NORDAT/FRQHOLD/TCODPRTY	
31-	1E	DISABLE SCRATCHPAD WRITE ENABLE	----
32-	1F	REPORT STATUS OF: ERROROK/CREGERR/AWBUSY/XMDFRZ	

NOTE: -In case of FUTURERR,NORDATA,ERROROK,CREGERR,AWBUSY & XMDFRZ
==== the registered outputs are reported, microcode has to take this fact into consideration. This applies to TEST CODES 1D & 1F only.
-FREEZE CODES with dashes implies that they are not needed and they are not known. (don't cares)

APPENDIX G - FREEZE CONTROL DIAGNOSTIC MODULE (Continued)

STATUS OF FREEZE SIGNALS DURING EACH TESTCODE:

CODES (01-17)(1D)(1F)
disable: OK2STOPH,EHOLD,EDFRZ,MODNEROK,MODEROKE,FETCHERR,HOLDAF HOLDBFPB.
enable : SPWENOK2/1 (if they are on),XMDERDIS.

CODES (0D)(1A-1C)
disable: no output signals.

CODES (1E)
disable: SPWENOK1/2

CODES (18-19)
disable: MODNEROK,MODEROKE,OK2STOPH,SPWENOK1/2 (for 2 clocks)
enable : EDFRZ,EHOLD (for 2 clocks and then disables them)
18 causes a 2 clock dead freeze.
19 causes a 2 clock live freeze.

CODES (1B)
inhibits the reception of all XMD related errors by turning on XMDERDIS and ignoring all returning XMD errors.