V-SERIES SPECIFICATIONS
DISTRIBUTION LIST
EMSG - MISSION VIEJO

SPECIFICATION NO: 1994 2408 SPECIFICATION TITLE: V500 EXEC MODULE MICROCODE

REVISION LETTER: A DATE OF ISSUE: 08-07-89

MISSION VIEJO, CA.

D.	ABALOS	MS 105
J.	BARRETT	MS 310
W.	BATES	MS 105
F.	BEUGER	MS 380
R.	BIGHAM	MS 195
D.	BRITTLE	MS 275
J.	DIEKMAN	MS 205
C.	DOOLING	MS 165
S.	ESCOBAR	MS 115
J.	GEORGE	MS 135
B.	GOULD	MS 175
F.	HAAS	
D.	HANSON	MS 205
R.	HARRIS	MS 135
A.	HATCH	MS 150
S.	HERMAN	MS 225
J.	HORKY	MS 170
M.	JACOBS	MS 165
M.	KALLAL	MS 105
C.	KING	MS 225
W.	LESH	RSM115
C.	LEUTHEN	MS 170
D.	MACMILLAN	MS 285
J.	MARTIN	MS 195
R.	MATHEWS	MS 225
V.	MCCLELLAND	MS 165
R.	MENDOZA	MS 135
J.	MOLASH	MS 135
B.	MORRIS	MS 135
W.	MURRAY	MS 165
M.	NAKAMOTO	MS 195
M.	OGRADY	MS 380
S.	PATRICK	MS 275
B.	POTTER	MS 135

CONTINUED - MISSION VIEJO, CA.

R.	QUAYLE	MS 135
J.	RUCKER	MS 520
J.	SINGLETON	MS 205
H.	SOHL	MS 165
S.	SOLOMON	MS 135
T.	STERNAL	MS 165
D.	STONE	MS 285
W.	TERRY	MS 215
T.	VECCHIARELLI	MS275 (2 COPIES)
D.	WALSH	MS 170
P.	WORCESTER	MS 135
J.	YETTER	MS 165

PASADENA, CA.

J.	BURTON	
N.	FLEMING	
R.	KATHOL	
C.	NUSE	
W.	TREINEN	CC 6130
J.	VANZWALUWENBURG	

TREDYFFRIN, PA.

A.	WORKS	

For changes in this distribution list, contact D. Patten, x5838, M.V. Engineering Documentation.

UNISYS CORPORATION	PAGE 1 OF 89	DOC SIZE A	DOC NO. 1994 2408	REV A
COMPUTER SYSTEMS GROUP MISSION VIEJO PLANT	DOC TYPE ES	CLASS CODE 4-0202	DESIGN CONT PLANT 037	

TITLE: V500 EXECUTE MODULE MICROCODE

REV	CONT DOC & NO.	ORIG	CHKD	APPD & DATE	REVISION DESCRIPTION
A	ECN 96015A	7/21/99		8/2/99	-INITIAL RELEASE-

Company Confidential

1994 2408

V500 EXECUTE MODULE MICROCODE

ENGINEERING DESIGN SPECIFICATION

UNISYS

GENERAL SYSTEMS GROUP

ORANGE COUNTY

TABLE OF CONTENTS

UNISYS CORPORATION
GENERAL SYSTEMS GROUP
MISSION VIEJO PLANT

1 PURPOSE

This document is intended to describe the microcode developed for the V500 EXECUTE Module (XM).

2 APPLICABLE DOCUMENTS

1993 5162	V500 System
1993 5170	V500 System Index
1993 5196	V500 Processor
1993 5279	V500 Architecture
1993 5204	V500 Execute Module
1993 5212	V500 Fetch Module
1993 5220	V500 Memory Controller & Cache Module
1993 5238	V500 Memory Data Card
1993 5246	V500 I/O Processor
1993 5303	V500 Maintenance Subsystem
1993 5295	V500 System Maintenance Controller
1997 5390	V Series Instruction Set
1993 5345	P5 Microcode Simulator "FXMULATOR"
1993 5113	V500 Microcode Compiler Program
(WIT78)	Design and Construction of Hierarchically Structured Software ", R.W. Witty, Atlas Computing Division, 1978

3 OVERVIEW

3.1 XM Hardware

The Execution Module (XM) processes data provided to it by input queues and interface, and send results, if needed, to other modules through its output queue and interface. It is made of a Data Section and Control Section.

3.1.1 Data Section

The Data Section's major components can be subdivided into three main parts:

the Storage Units,
the Interconnection Unit and
the Data Operators.

The Storage Units are subdivided into Queues and Local Storages.

The Input Queues capture data coming from other modules (IF, OF or MCACM), and consist of the FETCH PAGES, OPERAND QUEUE and READ BUS QUEUE.

The Output or WRITE QUEUE holds temporarely the outgoing data.

The Local Storages hold the information that is internal to XM for OP execution and consist of SCRATCHPAD, MASS STORE and registers such as AUREG, DREG, PTREG, MREQ's A, B and C register pairs.

The LITERAL FILE provides a set of literal words usable by the Data Section, it is a read only device.

The Inter Connection Unit performs all the data movement within XM; It is basically a crossbar switch.

Data manipulation is performed by the following units:

Arithmetic Unit (AU)
Logic Unit (LU)
Comparison Unit
Programmable Test Unit (PTEST)
Programmable Multiplexor Unit (PMUX)
Memory Requestor Unit (MREQ)

3.1.2 Control Section

The Control Section's major components are:

Control Store (CRAM) and its Control Register (CREG),
Microsequencer and its Stack,
FBUS2 Queue,
Timers and
State Registers.

3.1.3 Interfaces

XM communicates with other modules through 2 interfaces:

the IF/OF interface and
the MCACM interface.

3.1.3.1 IF/OF Interface

XM captures the information coming from the FETCH Modules through 2 busses called FBUS1 and FBUS2.

From XM, FBUS1 consists of a 10 digit wide bus feeding the " Data " part of the FETCH PAGE. It carries informations that will exclusively be handled in the XM DATA Section.

FBUS2 consists of a 10 bit wide bus feeding the " Control " part of the FETCH PAGE and feeds the XM CONTROL Section.

To complete the IF/OF-XM interface there exists a set of CONTROL lines Some of them are seen by the Microcode, others are only Hardware oriented.

3.1.3.2 MCACM Interface

The ADDRESS, READ and WRITE busses are all ten digit wide.

Along with the READ DATA bus there is an ERROR field which describes any error that is associated with the incoming data, and is saved in the ERROR Register.

The CONTROL Bus consists of a LENTGH field of 4 bits, a COMMAND field of 5 bits and 4 other wires which handles the interface handshake.

To complete the interface, there exists a TAG field which describes the destination module of the data that is transitting on the bus.

3.2 XM Microcode

Most of the XM Hardware is driven by microcode. A microcode word is 135 bit long. It has been designed to allow high parallelism in operations and data movement within the module.

The language used to develop it is specific to the V500 project. Its main characteristics are its structural simplicity, its ease of use and its short compilation time on BURROUGHS B7900. Refer to Appendix A for the Syntax description.

The XM microcode object is completely located in the Microcode Store , also called CONTROL STORE , which size is 16k words.

The microcode listing is made of 2 main parts : GLOBAL and MICROCODE STATEMENTS. GLOBAL contains all the DEFINE and FIELD descriptions used in the second part. Along with each symbolic microcode statement are 2 descriptions of the same microword. The first one is hardware related, bits are packed in 16 bit groups. The second one is firmware related, bits are grouped on a 'per function' basis. Refer to the PRINTCODE section of GLOBAL for details.

4 MAIN FEATURES OF XM OPERATIONS

4.1 GENERAL FLOW OF OPERATION

XM performs the execution of instructions using the information passed by IF/OF into the FETCH PAGES (up to 4) and other registers.

For performance purposes, OF pre-reads up to 10 digits of operand A and, if needed, operand B for XM.

This means that, in the "OPTIMAL" case, XM can count on having all that is needed for executing an OP, 'code' and 'data'.

In the "NON OPTIMAL" case, the pre-read data is disregarded and XM performs ALL the memory reads needed by the instruction.

If results have to be written back to memory, the data is sent to the WRITE queue which takes care of its transfer to MCACM.

4.2 OPLIT BRANCH

In order to eliminate the clocks that XM would have spent just to decode the opcode, check for a literal and an "optimal" situation , an initial 512 Way Case also called OPLIT BRANCH has been developed. The microaddress of the branch that follows every last clock of an instruction execution is built up with the following informations sent on the IF/OF interface:

OP Vector (7 bits from FBUS2),
LITERAL Flag (1 bit from FBUS2) and
OPTIMAL line .

Refer to section 5.2 for details.

4.3 CONDITIONAL READ

Most of the time, OF issues the operand pre-read commands before it is able to detect that the data pre-read can be the one that XM has not updated yet, that situation is called a DATA HIT. In order to invalidate the pre-read information, OF sends a "DATA HIT" signal to XM, notifying it that the incoming data will not be good.

That is why XM issues one or two CONDITIONAL READS, depending on the number of operands needed by the instruction, in the first few clocks of an OP execution. The read(s) will effectively go out only if the DATA HIT signal is active. This last test is performed by the hardware and is completely transparent to the firmware.

Refer to section 5.1 for details

4.4 RETRIABILITY

One of the main characteristics of the V500 is its fault tolerance, and one of its features is RETRIABILITY. Most of the instructions are retriable by the Maintenance Processor up to the point of a machine "STATE" modification, write to memory or processor state change. That point in time is tagged by the micro-code through the setting of a special Flip-Flop called "NOTRY" F/F.

Before the Maintenace Processor attempts to retry a failing execution it checks for the state of that F/F, if it is set, the instruction may not be retried.

4.5 WRITE ERROR REPORTING

The V500 is highly pipe-lined. That causes " asynchronism " between its modules. One of the problems related to that is the write error reporting. Indeed, XM does not wait a response from MCACM on the validity of the writes issued by it before performing further operations which can possibly relate to next instruction(s).

That is why the firmware issues a WRITE PC command before every first write related to the execution of an OP, so that if a write error is detected by MCACM, it can return the PC of the failing instruction to XM.

4.6 LOCK UNIT SYNCHRONISATION

'OF' module memorizes the addresses of all the data being accessed down the pipe - line in its LOCK UNIT. Whenever XM is finished with a piece of data which memory address space is described in the LOCK UNIT, it sends an OP COMPLETE signal to OF which in effect tells it that that specific data is now available for further use.

4.7 FAULT/INTERRUPT HANDLING

Unless it is caused by the XM firmware itself, the branch to the Fault or Interrupt Handlers is transparent to the microcode; indeed if at the time of the OPLIT branch the Interrupt line is active (an "asynchronous" fault and/ or interrupt is present) , the microsequencer forces the firmware to address '3FF0' Hex, label FLTINTASYNC.

This mechanism relieves the microcode from having to branch to a routine which would check for the presence of possible active asynchronous events that have to be served.

4.8 PIPE FLUSH

XM issues 2 types of Pipe Flush.

One, FETCH FLUSH, is for FETCH Modules only. It is needed to redirect the pipe when XM has detected a mispredicted BRANCH. Note that for correctly predicted BRANCHES , XM issues a BRANCH OK signal to FETCH.

The other one, SYSTEM FLUSH, is sent to all the modules. It is issued by XM to notify that a CONTEXT SWITCH (or ENVIRONMENT CHANGE) has been performed. A new BASE/LIMIT Table has been written and processing from the new context can start.

4.9 TIME OUT DETECTION

For some specific OP's or sub-routines which execution can lead to an infinite loop, XM (and also OF) microcode is able to detect a time out condition, in which case a branch to the FAULT HANDLER is performed.

Special care has been taken to cover all possible time out situations by micro-code because if it is not detected by the firmware, it will be by the hardware, and the action taken then is a DEAD FREEZE of the module.

Refer to section 5.11 for details.

4.10 SPECIAL SIGNALS HANDLING

There exist some special handshakes between FETCH and XM which are supported by the following signals : LIX OK, FETCH EVENT and XM EVENT.

LIX OK is issued by XM to notify FETCH that a Mobile Index Register Manipulation is being performed ; that allows FETCH to validate its cached copies of the registers.

FETCH EVENT and XM EVENT are used whenever one module has to wait for the other to finish a specific operation before it can itself proceed further.

4.11 STANDARD INSTRUCTION FLOW

```
|
.
.
GOTO5120 PAGESW;                    Last Micro Instruction
|                                   of Previous OP execution
|
|                                   <- OP X Execution Start
Micro Instruction #1                Located in the OPLIT TABLE
GOTO LABEL1;
|
|
Micro Instruction #2                LABEL1, if OPTIMAL, CREAD
|                                   is issued a.s.a.p.
.
.
.
Micro Instruction #n                includes SET NOTRY and
|                                   WRITE PC if followed by
|                                   Write to memory
|
Micro Instruction #n+1              includes WRITE to Memory
|
.
.
|
Micro Instruction #m                includes OP COMPLETE if
|                                   last write to memory
|
Last Micro Instruction              <- OP X Execution End
GOTO5120 PAGESW;
|
Micro Instruction #1                - OP 'Y' Execution Start
GOTO LABEL2;
or
Fault/Int. Handler Micro Inst. #1   - If the INTERRUPT Line is
|                                     active ( 3FF0H )
.
.
.
```

4.12 CONTROL STORE ADDRESS MAPPING

ADDRESS:LENGTH (HEX) (DEC)	LABEL	DESCRIPTION
0000:1	PAGESW	Reserved
0001:1		Microcode Date & Time Stamp
0002:1	SPCLOPOF	Reserved for OF "1C/1A/1E"
0003:1		Reserved
0004:252		OPLIT TABLE
0100:16		FAULT ENTRY TABLE
0110:16096		Body of Microcode
3FF0:2	FLTINTASYNC	Asynchronous Fault/Interrupt Interface Start
3FFD:1	FLTMC1	MCACM detected Fault Handling Start
3FFE:1	SETINTMOD	Interrupt Procedure Start
3FFF:1	FLTMCTEST	Fault Handler Start

5. DETAILED DESCRIPTION

5.1 "OPTIMAL" AND "OPERAND PREFETCH" - DEFINITION

Because it is critical from a performance point of view, a special care has been taken in defining the "OPTIMAL" signal and the kind of information that OF module pre-fetches for XM.

The following table describes it on an instruction basis.

SYNTAX DEFINITION

LEN-SYL	:	AFBF
A,B MOD 10	:	Both A and B operand MOD 10 read operand A into OPQ A, operand B into OPQ B.
A(xxxx)	:	Read A operand at absolute address xxxx into OPQ A.
[IXn]	:	Content of Index Register n
A* : MOD 10	:	If not literal then read length mod 10 from A address into OPQ A else no read.
48:1	:	read 1 digit from base 0 relative location 48.
a= A[xx:yy]	:	Read operand A[xx:yy] into OPQ A

OP CODE			OPTIMAL SET	LITERAL SET	OPERAND PREFETCH
#	MNEM	OPVEC	on ...	on ...	
ARITHMETIC INSTRUCTIONS; FIXED POINT, VARIABLE FIELD LENGTH					
01	INC	01	A,BLEN=<10	AF	A*,B MOD 10
02	ADD	02	"	"	"
03	DEC	03	"	"	"
04	SUB	04	"	"	"
05	MPY	05	"	"	"
06	DIV	06	"	"	"
ARITHMETIC INSTRUCTIONS; FIXED POINT, FIXED FIELD LENGTH					
50	IAD	28	RESET	RESET	a=A[7:8]
51	IAS	28	RESET	SET	"
52	ISU	28	SET	RESET	"
53	ISS	28	SET	SET	"
54	IMU	29	RESET	RESET	"
55	IMS	29	RESET	SET	"
57	IMI	29	SET	RESET	"
58	ILD	29	SET	SET	"
59	IST	2A	RESET	RESET	NONE
ARITHMETIC INSTRUCTIONS; FLOATING POINT, FIXED FIELD LENGTH					
70	RAA	2A	RESET	SET	a=A[11:4],b=A[7:8] (1) / a=A[19:10],b=A[9:10] (2)
71	RAS	2A	SET	RESET	"
72	RSU	2A	SET	SET	"
73	RSS	2B	RESET	RESET	"
74	RMU	2B	RESET	SET	"
75	RMS	2B	SET	RESET	"
76	RDV	2B	SET	SET	"
77	RDS	2C	RESET	RESET	"
78	RLD	2C	RESET	SET	"
79	RST	2C	SET	RESET	NONE
84	ACM	2C	SET	SET	NONE

(1) if Ac not equal to 01
(2) if Ac equal to 01

OP CODE			OPTIMAL SET on ...	LITERAL SET on ...	OPERAND PREFETCH
#	MNEM	OPVEC			
BRANCH INSTRUCTIONS					
21	LSSNN	2D	RESET	RESET	NONE
22	EQLNN	2E	"	"	"
23	LEQNN	2F	"	"	"
24	GTRNN	30	"	"	"
25	NEQNN	31	"	"	"
26	GEQNN	32	"	"	"
B1	LSSNT	2D	"	SET	"
B2	EQLNT	2E	"	"	"
B3	LEQNT	2F	"	"	"
B4	GTRNT	30	"	"	"
B5	NEQNT	31	"	"	"
B6	GEQNT	32	"	"	"
E1	LSSTN	2D	SET	RESET	"
E2	EQLTN	2E	"	"	"
E3	LEQTN	2F	"	"	"
E4	GTRTN	30	"	"	"
E5	NEQTN	31	"	"	"
E6	GEQTN	32	"	"	"
F1	LSSTT	2D	"	SET	"
F2	EQLTT	2E	"	"	"
F3	LEQTT	2F	"	"	"
F4	GTRTT	30	"	"	"
F5	NEQTT	31	"	"	"
F6	GEQNN	32	"	"	"
27	BUN	33	RESET	RESET	"
28	OFL	33	RESET	SET	"
2A	NUL	33	SET	SET	"
2B	GIN	34	RESET	RESET	"
20	NOP	34	RESET	SET	"

OP CODE			OPTIMAL SET on ...	LITERAL SET on ...	OPERAND PREFETCH
#	MNEM	OPVEC			
HALT INSTRUCTIONS					
29	HBR	33	SET	RESET	a=[48:1]
48	HBK	22	RESET	RESET	a=[46:3](B#0)
ENVIRONMENT CHANGE INSTRUCTIONS					
30	BCT	23	SET	RESET	NONE
31	NTR	23	SET	SET	a=IX3:8
32	EXT	24	RESET	RESET	a=IX3:8
35	VEN	1C	RESET	AF	a=B[19:10],b=B[9:10]
61	ASP	1B	SET	AF	a=A*:MOD10,b=[40:6](B#0)
62	HCL	1C	SET	AF	NONE
63	RET	24	RESET	SET	a=[[IX3]-6:10] b=[[IX3]+4:10]
90	INT	24	SET	RESET	NONE
93	BRV	24	SET	SET	NONE
DATA MOVEMENT INSTRUCTIONS					
08	MVD	21	RESET	RESET	NONE
09	MVL	20	ALEN=<10	RESET	A*,B MOD 10
10	MVA	07	A,BLEN=<10	AF	A*: MOD 10
11	MVN	08	A,BLEN=<10	AF	"
12	MVW	1E	LEN-SYL=<8	RESET	"
13	MVC	1E	LEN-SYL=<8	RESET	"
14	MVR	09	ALEN=<10	AF	"
15	TRN	21	RESET	SET	NONE
49	EDT	11	RESET	AF	NONE

OP CODE			OPTIMAL SET on ...	LITERAL SET on ...	OPERAND PREFETCH
#	MNEM	OPVEC			
LOGICAL INSTRUCTIONS					
16	SDE	12	A,BLEN=<10	AF	A*,B MOD 10
17	SDU	13	"	"	"
18	SZE	14	"	"	"
19	SZU	15	"	"	"
33	BRT	1F	ALEN=<10	RESET	A* : MOD 10
34	BST	1F	"	SET	"
37	SLL	16	"	AF	A*MOD10, B:6
38	SLD	17	"	"	"
39	SEA	18	"	"	A*,B MOD 10
40	BZT	19	"	"	A* : MOD 10
41	BOT	1A	"	"	"
45	CPA	0A	A,BLEN=<10	"	A*,B MOD 10
46	CPN	0B	"	"	"
42	AND	0C	"	"	"
43	ORR	0D	"	"	"
44	NOT	0E	"	"	"
64	SLT	21	SET	RESET	a=C[19:10], b=C[7:8]
66	STB	21	SET	SET	a=C[19:10], b=C[9:10]
INPUT/OUTPUT INSTRUCTIONS					
85	CIO	22	RESET	SET	a=A[19:10], b=A[9:10]
91	SRD	22	SET	RESET	a=AFBF[7:8]
92	RAD	22	SET	SET	NONE
94	IIO	1D	RESET	AF	a=A[19:10], b=A[9:10]
95	RDT	23	RESET	RESET	NONE
97	STT	23	RESET	SET	a=A[19:10], b=A[9:10
98	IOC	1D	SET	AF	a=A[5:6]

OP CODE			OPTIMAL SET	LITERAL SET	OPERAND PREFETCH
#	MNEM	OPVEC	on ...	on ...	
BINARY/DECIMAL CONVERSION INSTRUCTIONS					
88	D2B	0F	ALEN<=10	AF	A*: MOD 10
89	B2D	10	"	"	"
MEASUREMENT INSTRUCTION					
87	MOP	1B	RESET	AF	a=A[5:6], b=B[5:6]
MISCELLANEOUS INSTRUCTIONS					
47	SMF	25	SET	SET	NONE
60	LOK	26	RESET	RESET	a=A[19:10], b=A[9:10]
65	WHR	26	RESET	SET	a=A[8:9]
67	LIX	26	SET	RESET	a= [7:8]
68	SIX	26	SET	SET	NONE
69	ILS	11	SET	AF	a=A[5:6]
6A	MLS	27	SET	SET	a=A[9:10], b=A[19:10]
86	ATE	27	RESET	RESET	a=A[7:8], b=B[7:8]
99	SST	27	RESET	SET	NONE
AB	BAD	27	SET	RESET	NONE
STRING INSTRUCTIONS					
A0	MVS	25	RESET	RESET	a=A[33:10], b=B[33:10]
A1	CPS	25	RESET	SET	a=A[33:10], b=B[33:10]
A2	HSH	25	SET	RESET	a=A[33:10], b=A[23:10]

5.2 OPLIT TABLE

The OPLIT TABLE address is assembled as follows:

```
MSB  OPVEC[6]
     OPVEC[5]
     OPVEC[4]
     OPVEC[3]
     OPVEC[2]
     OPVEC[1]
     OPVEC[0]
     OPTIMAL
LSB  LITERAL
```

Next is the detailed description of the table itself.

Entries in it are classified on an Instruction basis: first the Ops with 4 entries (all possible combinations of OPTIMAL and LITERAL) then the Ops with 2 entries (state of OPTIMAL or LITERAL) and finally the Ops with 1 entry for which OPTIMAL and LITERAL are invalid.

Note on label naming convention:

```
XXXNONL : OpCode Mnemonic "XXX", No Optimal, No Literal
XXXONL  :           "            Optimal, No Literal
XXXNOL  :           "            No Optimal, Literal
XXXOL   :           "            Optimal, Literal
XXXNN   :           "            Not Taken, Not Taken
XXXTT   :           "            Taken, Taken
XXXNT   :           "            Not Taken, Taken
XXXTN   :           "            Taken, Not Taken
```

OP VECTOR (HEX)	OP TABLE ADDRESS (HEX)	OP DESCRIPTION (DECIMAL)	OP DESCRIPTION LABEL
00	00		Reserved
	01		(1)
	02		(2)
	03		Reserved
01	04	01	INCNONL
	05		INCNOL
	06		INCONLL
	07		INCOL
02	08	02	ADDNONL
	09		ADDNOL
	0A		ADDONL
	0B		ADDOL
03	0C	03	DECNONL
	0D		DECNOL
	0E		DECONL
	0F		DECOL
04	10	04	SUBNONL
	11		SUBNOL
	12		SUBONL
	13		SUBOL
05	14	05	MPYNONL
	15		MPYNOL
	16		MPYONL
	17		MPYOL
06	18	06	DIVNONL
	19		DIVNOL
	1A		DIVONL
	1B		DIVOL
07	1C	10	MVANONL
	1D		MVANOL
	1E		MVAONL
	1F		MVAOL
08	20	11	MVNNONL
	21		MVNNOL
	22		MVNONL
	23		MVNOL

(1) : Date and Time Stamp
(2) : Reserved for "1A, 1C, 1E" OP's

OP VECTOR (HEX)	OP TABLE ADDRESS (HEX)	OP DESCRIPTION (DECIMAL)	OP DESCRIPTION LABEL
09	24	14	MVRNONL
	25		MVRNOL
	26		MVRONL
	27		MVROL
0A	28	45	CPANONL
	29		CPANOL
	2A		CPAONL
	2B		CPAOL
0B	2C	46	CPNNONL
	2D		CPNNOL
	2E		CPNONL
	2F		CPNOL
0C	30	42	ANDNONL
	31		ANDNOL
	32		ANDONL
	33		ANDOL
0D	34	43	ORRNONL
	35		ORRNOL
	36		ORRONL
	37		ORROL
0E	38	44	NOTNONL
	39		NOTNOL
	3A		NOTONL
	3B		NOTOL
0F	3C	88	D2BNONL
	3D		D2BNOL
	3E		D2BONL
	3F		D2BOL
10	40	89	B2DNONL
	41		B2DNOL
	42		B2DONL
	43		B2DOL
11	44	49	EDTNONL
	45		EDTNOL
	46	69	ILSNL
	47		ILSL

OP VECTOR (HEX)	OP TABLE ADDRESS (HEX)	OP DESCRIPTION (DECIMAL)	OP DESCRIPTION LABEL
12	48	16	SDENONL
	49		SDENOL
	4A		SDEONL
	4B		SDEOL
13	4C	17	SDUNONL
	4D		SDUNOL
	4E		SDUONL
	4F		SDUOL
14	50	18	SZENONL
	51		SZENOL
	52		SZEONL
	53		SZEOL
15	54	19	SZUNONL
	55		SZUNOL
	56		SZUONL
	57		SZUOL
16	58	37	SLLNONL
	59		Invalid
	5A		SLLONL
	5B		SLLOL
17	5C	38	SLDNONL
	5D		Invalid
	5E		SLDONL
	5F		SLDOL
18	60	39	SEANONL
	61		Invalid
	62		SEAONL
	63		SEAOL
19	64	40	BZTNONL
	65		Invalid
	66		BZTONL
	67		BZTOL
1A	68	41	BOTNONL
	69		Invalid
	6A		BOTONL
	6B		BOTOL

OP VECTOR (HEX)	OP TABLE ADDRESS (HEX)	OP DESCRIPTION (DECIMAL)	OP DESCRIPTION LABEL
1B	6C	87	MOPNL
	6D		MOPL
	6E	61	ASPNL
	6F		ASPL
1C	70	35	VENNL
	71		VENL
	72	62	HCLNL
	73		HCLL
1D	74	94	IIONL
	75		IIOL
	76	98	IOCNL
	77		IOCL
1E	78	12	MVWNO
	79	13	MVCNO
	7A	12	MVWO
	7B	13	MVCO
1F	7C	33	BRTNO
	7D	34	BSTNO
	7E	33	BRTO
	7F	34	BSTO
20	80	09	MVLNO
	81		Reserved
	82	09	MVLO
	83		Reserved
21	84	08	MVD
	85	15	TRN
	86	64	SLT
	87	66	STB
22	88	48	HBK
	89	85	CIO
	8A	91	SRD
	8B	92	RAD
23	8C	95	RDT
	8D	97	STT
	8E	30	BCT
	8F	31	NTR

OP VECTOR (HEX)	OP TABLE ADDRESS (HEX)	OP DESCRIPTION (DECIMAL)	OP DESCRIPTION LABEL
24	90	32	EXT
	91	63	RET
	92	90	INT
	93	93	BRV
25	94	A0	MVS
	95	A1	CPS
	96	A2	HSH
	97	47	SMF
26	98	60	LOK
	99	65	WHR
	9A	67	LIX
	9B	68	SIX
27	9C	86	ATE
	9D	99	SST
	9E	AB	BAD
	9F	6A	MLS
28	A0	50	IAD
	A1	51	IAS
	A2	52	ISU
	A3	53	ISS
29	A4	54	IMU
	A5	55	IMS
	A6	57	IMI
	A7	58	ILD
2A	A8	59	IST
	A9	70	RAA
	AA	71	RAS
	AB	72	RSU
2B	AC	73	RSS
	AD	74	RMU
	AE	75	RMS
	AF	76	RDV
2C	B0	77	RDS
	B1	78	RLD
	B2	79	RST
	B3	84	ACM

OP VECTOR (HEX)	OP TABLE ADDRESS (HEX)	OP DESCRIPTION (DECIMAL)	OP DESCRIPTION LABEL
2D	B4	21	LSSNN
	B5	B1	LSSNT
	B6	E1	LSSTN
	B7	F1	LSSTT
2E	B8	22	EQLNN
	B9	B2	EQLNT
	BA	E2	EQLTN
	BB	F2	EQLTT
2F	BC	23	LEQNN
	BD	B3	LEQNT
	BE	E3	LEQTN
	BF	F3	LEQTT
30	C0	24	GTRNN
	C1	B4	GTRNT
	C2	E4	GTRTN
	C3	F4	GTRTT
31	C4	25	NEQNN
	C5	B5	NEQNT
	C6	E5	NEQTN
	C7	F5	NEQTT
32	C8	26	GEQNN
	C9	B6	GEQNT
	CA	E6	GEQTN
	CB	F6	GEQTT
33	CC	27	BUN
	CD	28	OFL
	CE	29	HBR
	CF	2A	NUL
34	D0	2B	GTN
	D1	20	NOP
	D2	XY	CHK
	D3 to FF		Reserved for Expansion

5.3 MICROCODE DATA STRUCTURES

There exist a series of "DATA STRUCTURES" that the Firmware often uses, the main ones are:

- MACHINE STATES,
- MASS STORE,
- BASE/LIMIT Table and
- SHARED MEMORY AREA.

5.3.1 MACHINE STATES

The MACHINE STATES is a collection of registers, scattered throughout XM, that is assembled into a frame. It is pushed to or popped from 'a' stack during ENVIRONMENT CHANGES (see section 5.8). It is described as follows:

- Accumulator (28 digits, in Mass Store),
- Measurement Register (8 digits, in Mass Store),
- Interrupt Mask (2 digits, discrete),
- Mobile Index Registers (32 digits, in Mass Store),
- Mode Indicators (2 digits, discrete),
- COMS & OVF Flags (2 digits, discrete),
- Environment Number (6 digits, in Mass Store),

5.3.2 MASS STORE

The Mass Store is a 256 word (10 digit) memory array local to XM, its usage is twofold.

The lower address space is a single port read, single port write, sequential access RAM area which serves as an extension to the Scratchpad . It spans from address '00' Hex to '8F' Hex.

Its higher address space contains various pointers, descriptors and registers that are cached locally in XM. It spans from address '90' Hex to 'FF' Hex, its access is mostly read only

5.3.2.1 MASS STORE ADDRESS MAPPING

ADDRESS (HEX)	MNEMONIC	DESCRIPTION	PATTERN (DGT)
00 to 8F	---	Scratchpad Area	---
90 to 94	Reserved		
95	SINTMASK	Saved Interrupt Mask Register	00000000MM
96	---		
97	MERROREP	MCACM Error Report	xxDDDDDDDD
98	MERRORADDR	MCACM Error Address	xxDDDDDDDD
99	DATE	Date	YYYYMMDDxx
9A	TIMESTART (11)	Task Timer Start Value	00TTTTTTTT
9B	ACCUMTIME (11)	Accumulated Time	TTTTTTTTTT
9C	KETA (1)	Kernel Environment Table Address	xAAAAAAAAA
9D	KETMAXE (1)	" Env. Tab. Max. # Entries	0000NNNNNN
9E	KSMATA (1)	" Services MAT Address	xAAAAAAAAA
9F	KSMATMAXE(1)	Services MAT Max. # Entries	0000NNNNNN

"x" in the Pattern column means "Don't Care"

ADDRESS (HEX)	MNEMONIC		DESCRIPTION	PATTERN (DGT)
A0	RLPB	(1)	Reinstate List Pointer Base	0AAAAAAAAA
A1	MCPETA	(1)	MCP Env. Tab. Add.	0AAAAAAAAA
A2	MCPETMAXE	(1)	MCP ENV. Tab. Max. # Entries	0000NNNNNN
A3	USMATA	(2)	User Services MAT Address	xAAAAAAAAA
A4	USMATMAXE	(2)	" MAT Max. # Ent.	00000000NN
A5	ATRLP	(2)	Active Task Reinstate List Pointer	xAAAAAAAAA
A6	ATN	(2)	" Number	xxxxxxNNNN
A7	ATETA	(2)	" Env. Table Address	FAAAAAAAAA
A8	ATETMAXE	(2)	" " Max. # Ent.	0000NNNNNN
A9	AEN		Active Environment Number	NNNNNN0000
AA	BA		Branch Address	xxxxAAAAAA
AB	FLAGS	(10)	IPC,MP,OVRTMP,BRV,IP,HCP,FF01	ABxxxCDEFG
AC	LIM0REL	(4)	Limit # 0, Base # 0 Relative	0000AAAAAA
AD	LIM1REL	(4)	Limit # 1, Base # 1 Relative	1000AAAAAA
AE	SEN		Saved Environment #	NNNNNN0000
AF	BLTABPNTR	(3)	B/L Table Desc. Pointer	V0000000Bn;n=0,1,2
B0	BLADESC	(3-4)	B/L Table A Desc.	EEEEEExxAA;AA= Bin. Adr
B1	BLBDESC	(3-4)	" B	
B2	BLCDESC	(3-4)	" C	
B3	SGNACCUM	(2-7)	Sign Accumulator	SEES000000
B4	MSWACCUM	(2-7)	Most Significant Word Accumulator	NNNNNNNN00
B5	LSWACCUM	(2-7)	Least Significant Word Accumulator	NNNNNNNN00
B6	EXTACCUM	(2)	Extension - Accumulator	0000000000
B7	MEAR	(2-8)	Measurement Register	MMMMMMMMxx
B8	IX4	(2-9)	Mobile Index Register # 4	IIIIIIIIxx
B9	IX5	(2-9)	" 5	IIIIIIIIxx
BA	IX6	(2-9)	" 6	IIIIIIIIxx
BB	IX7	(2-9)	" 7	IIIIIIIIxx
BC	---			
BD	MEMDESC	(3)	Memory Descriptor	xxxxxxxTTN
BE	PROCNMBR	(3)	Processor Number	000000N000
BF	NFTLERVEC		Non Fatal Error Vector	00000000NN

"x" in the Pattern column means "Don't Care"

ADDRESS (HEX)	MNEMONIC		DESCRIPTION	PATTERN (DGT)
C0	B0A	(4)	Base # 0	0NNNNNN000
C1	L0A	.	Limit # 0	0NNNNNN000
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
CE	B7A	.	Base # 7	7NNNNNN000
CF	L7A	.	Limit # 7	7NNNNNN000
D0	B0B	(4)	Base # 0	same as
D1	L0B	.	Limit # 0	above
.	.	.	.	
.	.	.	.	
.	.	.	.	
DE	B7B	.	Base # 7	
DF	L7B	.	Limit # 7	
E0	B0C	(4)	Base # 0	same as
E1	L0C	.	Limit # 0	above
.	.	.	.	
.	.	.	.	
.	.	.	.	
EE	B7C	.	Base # 7	
EF	L7C	.	Limit # 7	
F0	B0K	(3)	Base # 0 Kernel	same as
F1	L0K	.	Limit # 0 Kernel	above
.	.	.	.	
.	.	.	.	
.	.	.	.	
FE	B7K	.	Base # 7 Kernel	
FF	L7K	.	Limit # 7 Kernel	

Notes on 5.3.2.1

(1) Updated during WHR
(2) Updated during BRV
(3) Loaded during INIT
(4) Updated during LOAD MAT
(5) "V" changed to Invalid (0) at ATE, WHR, IP, BRV
(6) Updated during LOAD MAT if needed
(7) Updated during ACCUM. OP's
(8) Updated during MOP
(9) Updated during LIX
(10) A = "F" if IPC Int On
B = "F" if MP Int On
C = "1" if OVERTEMP INT MASK is On
D = "F" if BRV Op being executed
E = "F" if Int.Proc. being executed
F = "F" if HCP being executed
G = FF10 Status
(11) Updated during STT, RDT, Interrupt and BRV

5.3.3 BASE LIMIT TABLE

All memory accesses are performed within a memory area which is described by a Base/Limit pair. The memory area selection is done via a Base Indicant (BI) digit which is the most significant digit of the address word.

The Base/Limit (B/L) Table is a 16 entry memory array residing in MCACM, it contains the current or in use memory area descriptors.

Base/Limit 0 through 7 are the ones software can reference to via indexed addresses. Base/Limit 8 through F have a very restricted usage, they are not referenced through the normal instruction set, they are for hardware use only.

Note that although addresses are like other data expressed in decimal, MCACM converts and stores internally the BASE and LIMIT value in binary.

5.3.3.1 B/L TABLE DESCRIPTION

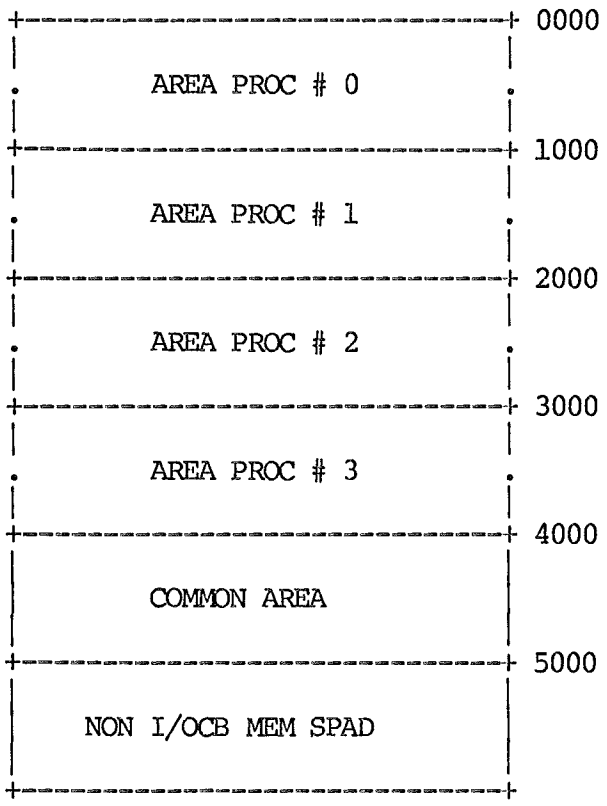
BI = 0, B/L Pair 0
" = 1, " 1
" = 2, " 2
" = 3, " 3
" = 4, " 4
" = 5, " 5
" = 6, " 6
" = 7, " 7
" = 8, Reserved - Multi Purpose, has to be reset at end of OP
" = 9, "
" = A, "
" = B, "
" = C, Reserved for " QWIK DISK " Base/Limit Pair
" = D, Reserved for Multiprocessor Shared Area Base/Limit Pair
" = E, Reserved for " MCP DATA AREA " Base/Limit Pair (1)
" = F, Reserved for " ABSOLUTE ADDRESS " Base/Limit Pair (2)

(1) Updated during BRV
(2) Updated during INIT

5.3.4 SHARED MEMORY AREA

One of the main characteristics of the V500 system is that it is a MULTIPROCESSOR. This means that although at any given time a processor works on a piece of data that is unique to itself , it can also access some data that is global to it and up to three other companions.

That is the reason for the existence of a SHARED MEMORY AREA (SMA). Its Base is greater than the Limit of the Operating System's address space and its Limit is the physical Limit of the memory. It is accessed via BASE/LIMIT Pair "D" and is laid out as follows:



The "AREA PROC # n" is dedicated to each individual Processor and includes info such as:

MOBILE INDEX REGISTERS (40 Digits)	
IX4	[0:8]
IX5	[10:8]
IX6	[20:8]
IX7	[30:8]
MODE DESCRIPTOR	[40:1]
IPC INFO	[41:NN]
COPROCESSOR INFO	[50:MM]

The "COMMON AREA" includes info such as:

KERNEL LOCK	[0:1]	- '0' = Available
RESERVED	[1:1]	
SNAP LOCK	[2:1]	- '0' = Available
SNAP PICTURE ENABLE	[3:1]	- 'Non Zero' = Set
M.E.R. LOCK	[4:2]	- '0' = Available
M.E.R. ENABLE	[6:2]	- 'Non Zero' = Set
Reserved	[8:2]	
HARDWARE REGISTERS (50 Digits)		
REINSTATE LIST ADDRESS	[10:9]	
SNAP PICTURE ADDRESS	[20:9]	
MEM. AREA STATUS TABLE ADDRESS	[30:9]	
MEM. ERROR REPORT ADDRESS	[40:9]	
RESERVED	[50:50]	
TIME OF DAY - Year/Month/Day part	[100:10]	

SYSTEM I/D	[112:200]
OVERTEMPERATURE Flag	[312:2] (Off = 00, On = Non Zero)
SYSTEM STATUS (10 Digits)	
ECM Failure	[314:2] (Off = 00, On = 20)
ECM Log Full	[316:2] (Off = 00, On = 10)
SNAP Picture Status	[318:2] (Off = 00, On = 04)
Temperature Warning	[320:2] (Off = 00, On = 02)
Voltage Warning	[322:2] (Off = 00, On = 01)
MER Status	[324:2] (Off = 00, On = 08)
RESERVED	[326:NN]
NON IOCB MEMORY SCRATCHPAD	[5000:5000]

5.4 EXCEPTION HANDLING

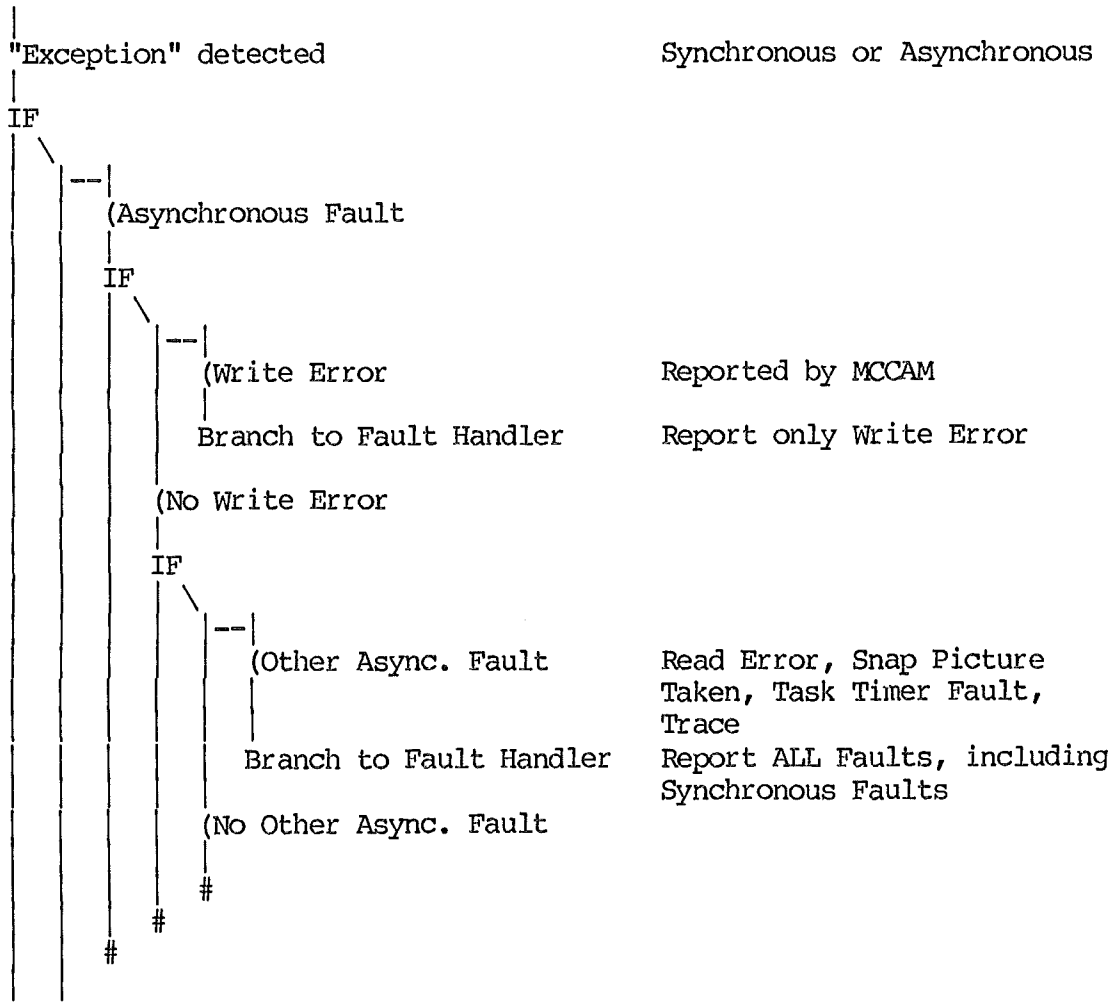
Events asynchronous to XM microcode that demand an interruption of the normal flow of OP execution are latched by the hardware in the INTERRUPT REGISTERS and are serviced at OPLIT BRANCH time.

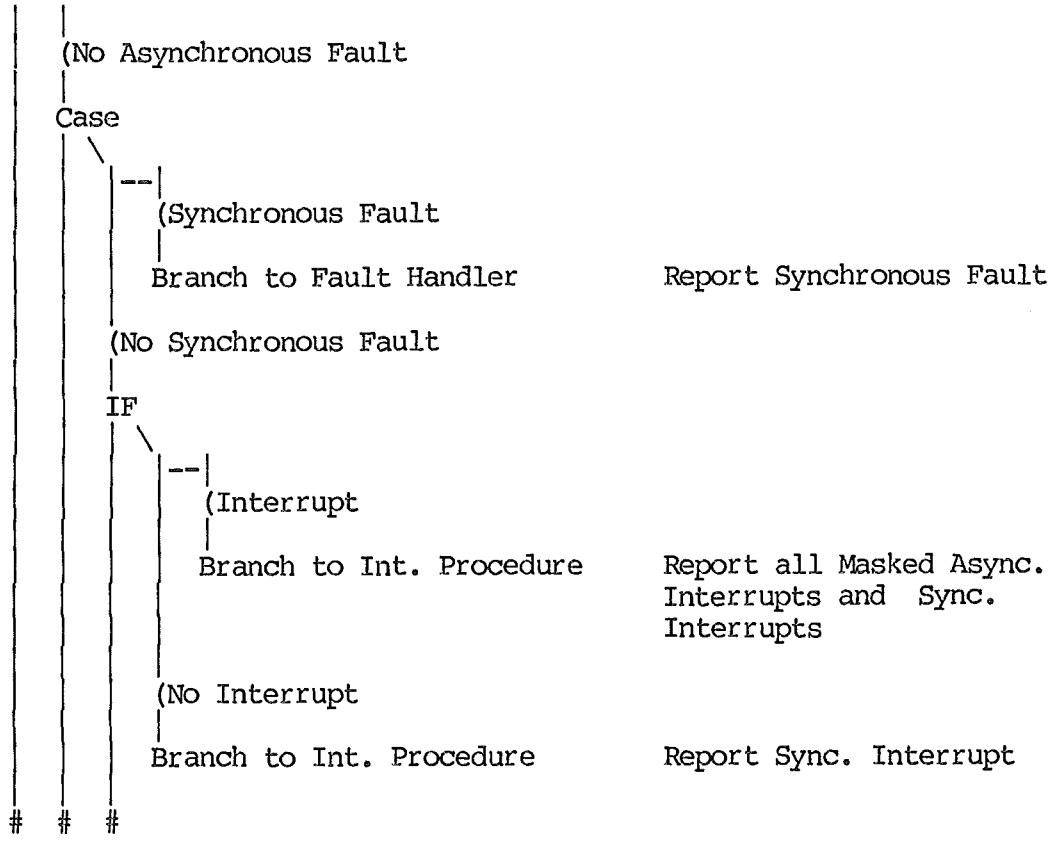
Events,or exception conditions, that are detected by the XM microcode are serviced immediately.

Because of the action they require, events or exceptions are divided into FAULTS and INTERRUPTS; the FAULTS trigger the FAULT HANDLER, the INTERRUPTS trigger the INTERRUPT HANDLER.

Since FAULTS describe that "something is wrong", they have a higher priority service than INTERRUPTS.

5.4.1 EXCEPTION SERVICING - PRIORITY SCHEME

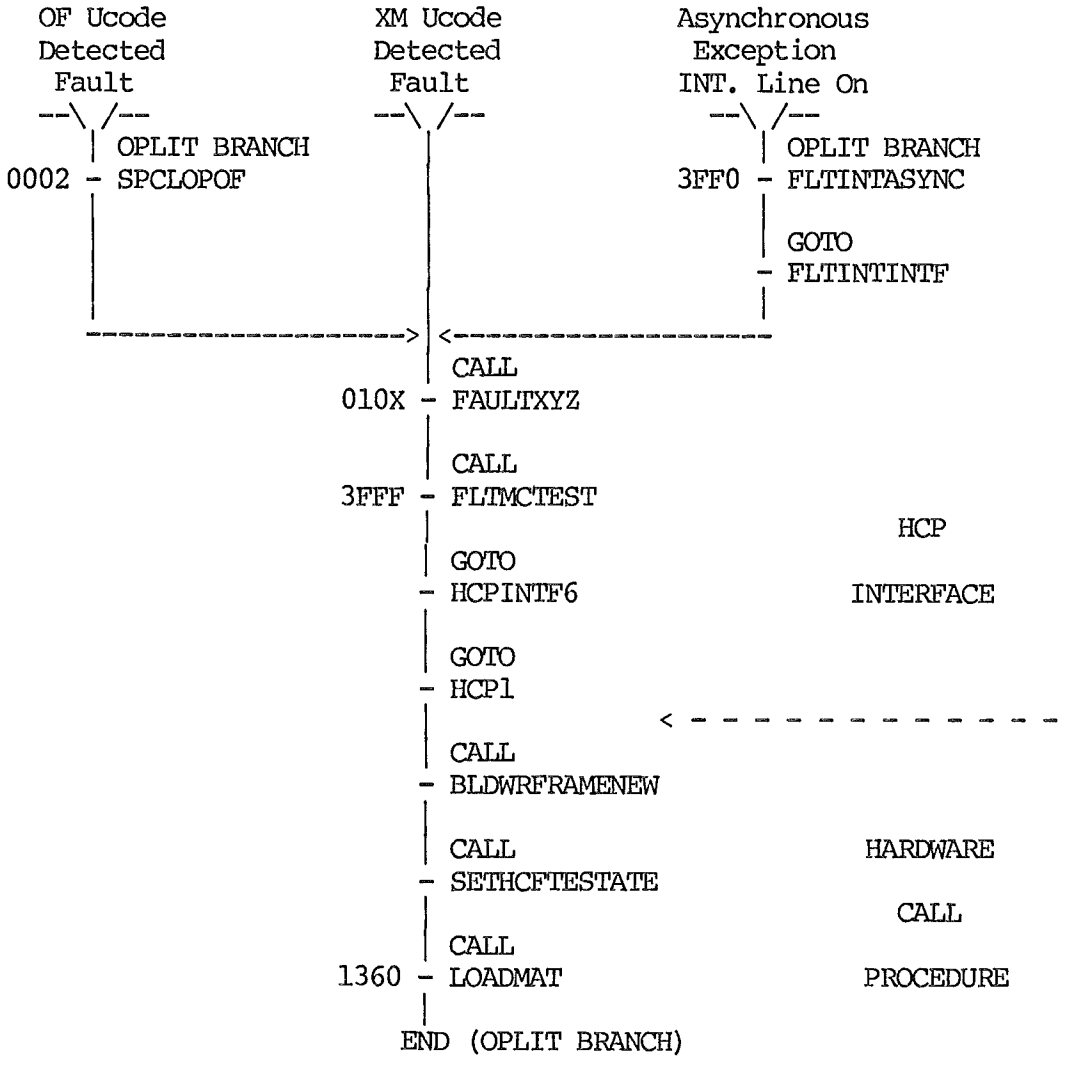



(No Asynchronous Fault
Case
(Synchronous Fault
Branch to Fault Handler
Report Synchronous Fault
(No Synchronous Fault
IF
(Interrupt
Branch to Int. Procedure
Report all Masked Async. Interrupts and Sync. Interrupts
(No Interrupt
Branch to Int. Procedure
Report Sync. Interrupt

5.4.2 FAULT HANDLER

The FAULT HANDLER is made of 2 main parts, the HCP INTERFACE and the HARDWARE CALL PROCEDURE (HCP).

The Fault Handler has 3 different entry points. Next is a flow diagram describing the main operations involved, they are tagged by their microaddress, label and the way they are reached (GOTO or CALL).



5.4.2.1 FAULT TABLE

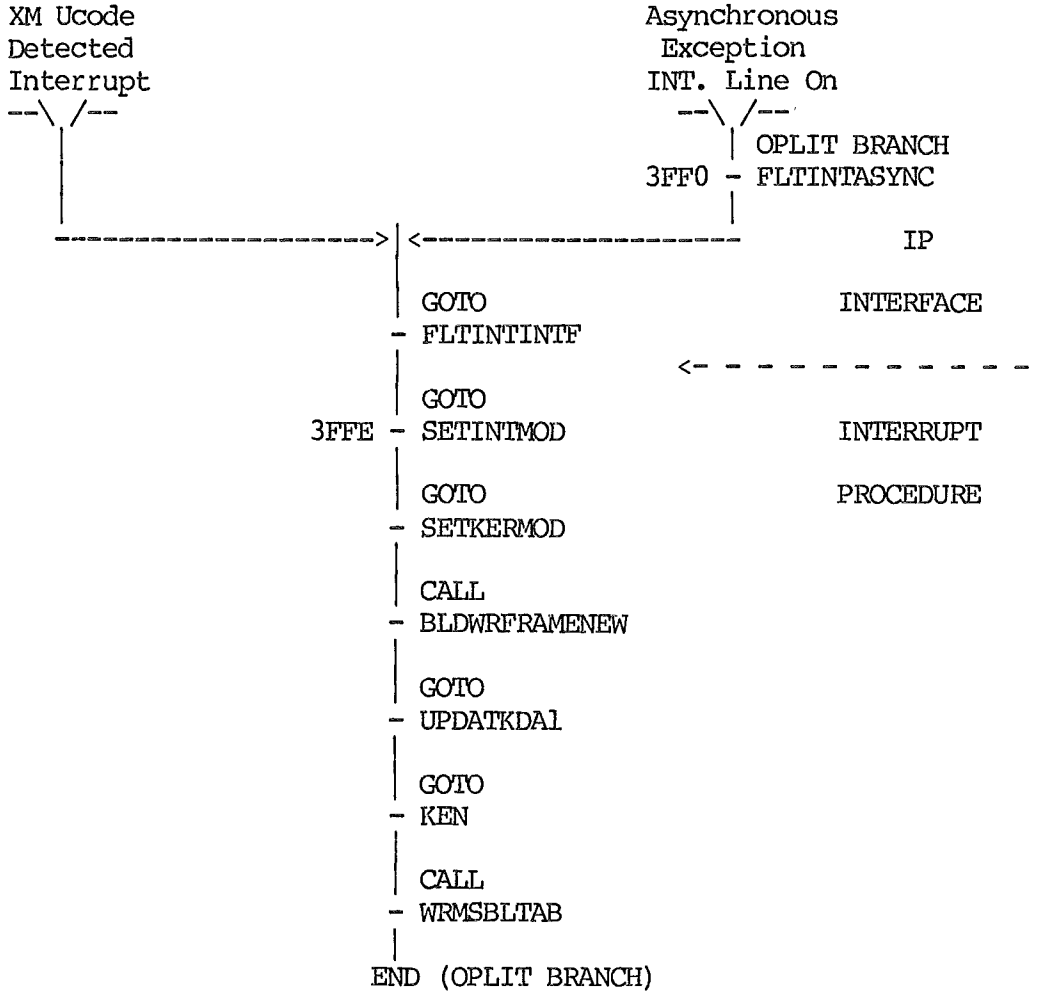
When a Fault is detected by the XM microcode, a Call of one of the following labels is performed.

Micro Address	Label	Description
0100	FAULTHMA	HARD MEMORY AREA
0101	FAULTT	TRACE
0102	FAULTIAD	INVALID ARITH. DATA
0103	FAULTSMA	SOFT MEMORY AREA
0104	FAULTII	INVALID INSTRUCTION
0105	FAULTUMPE	UNCORRECTABLE MEMORY
0106	FAULTAE	ADDRESS ERROR
0107	FAULTIT	INSTRUCTION TIMEOUT
0108	FAULTSO	STACK OVERFLOW
0109	FAULTAT	ACCUMULATOR TRAP
010A	Reserved	---
010B	FAULTSF	SOFT FAULT
010C	FAULTSKT	TASK TIMER = 0
010D	UMPE	MP Entry - Restart XM after Double Bit Parity Error detected
010E	Reserved	---
010F	SNAPTAKEN	MP Entry - Restart XM after SNAP Pict. Stored

5.4.3 INTERRUPT HANDLER

The INTERRUPT HANDLER is made of two main parts, the IP INTERFACE and the INTERRUPT PROCEDURE (IP).

The Interrupt Handler has 2 different entry points. Next is a flow diagram describing the main operations involved, they are tagged by their microaddress, label and the way they are reached (GOTO or CALL).



5.5 MEMORY ADDRESSING

Unless it is accessed via an 'ABSOLUTE ADDRESS', a data in memory is always base relative. The base is described by the most significant digit of the address.

A MEMORY AREA is defined by a Base and a Limit. A collection of related Memory Areas is called a MEMORY AREA TABLE (M.A.T.).

A MAT is what describes an ENVIRONMENT. A collection of related Environments is called an ENVIRONMENT TABLE (E.T.).

An Environment Table describes a TASK. The collection of runnable Tasks is described in a table called the REINSTATE LIST.

All the pointers to these Data Structures are registered within the Mass Store - Address 'A0' to 'A9'.

Refer to section 5.3.2.1 for details.

5.5.1 ENVIRONMENT CHANGE

When an Active TASK needs to access data from another Environment, which is not currently described, it triggers what is called an ENVIRONMENT CHANGE. That is translated by XM microcode into the execution of a routine called LOADMAT, label LOADMAT - microaddress '1360' Hex, which consists of resolving the target Environment's M.A.T. into a new BASE/LIMIT Table. When that operation is finished, XM sends a SYSTEM FLUSH; this signals the other modules, which are generally idle at that time, that processing from the new Environment can begin.

5.5.2 BASE/LIMIT TABLE CACHE

The resolution of a Memory Area Table (MAT) can be time consuming, it depends on the number of indirection that are encountered.

Various features have been developped to help its processing.

The registering in Mass Store of heavily used pointers is one. The pointers are:

```
        the MCP ET Address            (location 'A0'),
        the MCP ET Size               (   "     'A1'),
        the User Services MAT Address (   "     'A2'),
        The User Services MAT Size    (   "     'A3')
```

In spite of this feature, it takes anywhere from 80 to 200 clocks or more to resolve an Environment. As a mean to decrease further the time spent in the process, a Base/Limit Table Cache has been developped. It is based on the assumption that the most recently resolved table is the one that is most likely to be used again.

The LOADMAT routine, before starting to resolve the target Environment Number, checks if the environment is already described in Mass Store. If so, it takes the Base/Limit table from it and sends it to MCACM. The whole routine takes then only about 35 clocks, irrespective of the MAT complexity.

The Mass Store words located at address AF,B0,B1 and B2 are the main components of the mechanism. They are described as follows:

Location AF : VxxxxxxxBn n = 0, 1 or 2

'V' is a validity digit, 'F' = valid,

'Bn' is a Mass Store Hex address pointing to the current 'Base/Limit Table' Descriptor.

Location Bn : EEEEEExxm0; m = C, D or E

Base/Limit Table Descriptor where

'EEEEEE' is an Environment Number and

'm0' is a Mass Store Hex address pointing to the beginning of the current Base/Limit Table.

Up to three Environments can be described at any given time.

A fourth Base/Limit table, which is independent of the pointers described above resides in the same memory array: it relates to the KERNEL MAT (Mass Store Address 'F0' for 16). It is different in nature from the other tables since its content does not change in time. This table is used every time an Interrupt forces the processor to go into the KERNEL.

Refer to section 5.3.2.1 for details.

Note that a Base or Limit word in Mass Store is described as 'INNNNNN000', where 'I' is a Base Indicant and NNNNNN is a decimal value.

5.6 TRACE HANDLING

The TRACE feature allows the software to have a visibility over the machine STATES after the execution of an OP.

TRACE is considered and treated as an ASYNCHRONOUS FAULT and its action is taken at OPLIT BRANCH time.

When an OP, generally RET from HCP/HCL, changes the TRACE ENABLE Mode Indicator from OFF to ON , the TRACE INTERRUPT signal that latches the change of the mode indicator, does not become active immediately. Its update occurs only at the end of the current op execution. This means that , assuming that no exception exists, the next OP is executed.

When the execution of this next OP finishes, since the INTERRUPT line is active, an automatic branch to microaddress '3FF0' Hex occurs, this starts the FAULT HANDLER which pushes the STATES of the machine and TRACE parameters into the HARDWARE CALL STACK.

If TRACE FAULT exists with some other FAULT(s) condition(s), all of them are reported in the FAULT DESCRIPTOR.

If TRACE FAULT coexists with some INTERRUPT(s), the FAULT is served first, the service of the interrupt is postponed except in one instance, i.e. a SYNCHRONOUS INTERRUPT caused by the LOCK instruction, variant 1 (Unconditional Lock),in which case the SYNCHRONOUS INTERRUPT is served before the TRACE FAULT.

5.7 TRAP HANDLING

The TRAP feature allows the software to have a visibility over the machine STATES when "something wrong" occured during the execution of ACCUMULATOR Op's. The "something wrong" can be EXPONENT OVERFLOW, EXPONENT UNDERFLOW and DIVIDE BY ZERO.

TRAP is considered and treated as a SYNCHRONOUS FAULT and its action is taken at the time the exception condition is detected.

5.8 SNAP HANDLING

For some FAULT conditions such as INVALID ARITHMETIC DATA, INSTRUCTION TIME OUT, ADDRESS ERROR and INVALID INSTRUCTION the software has an expanded visibility over the hardware by being able to make a SNAP PICTURE of additional memory elements of a processor.

Since there exists only one SNAP AREA for up to 4 processors, the SMA is used to sequentially discriminate the possible requestors through the SNAP LOCK, COMMON AREA [2:1] (refer to section 5.3.4).

The taking of a SNAP PICTURE requires a handshake between XM and MP that works as follows:

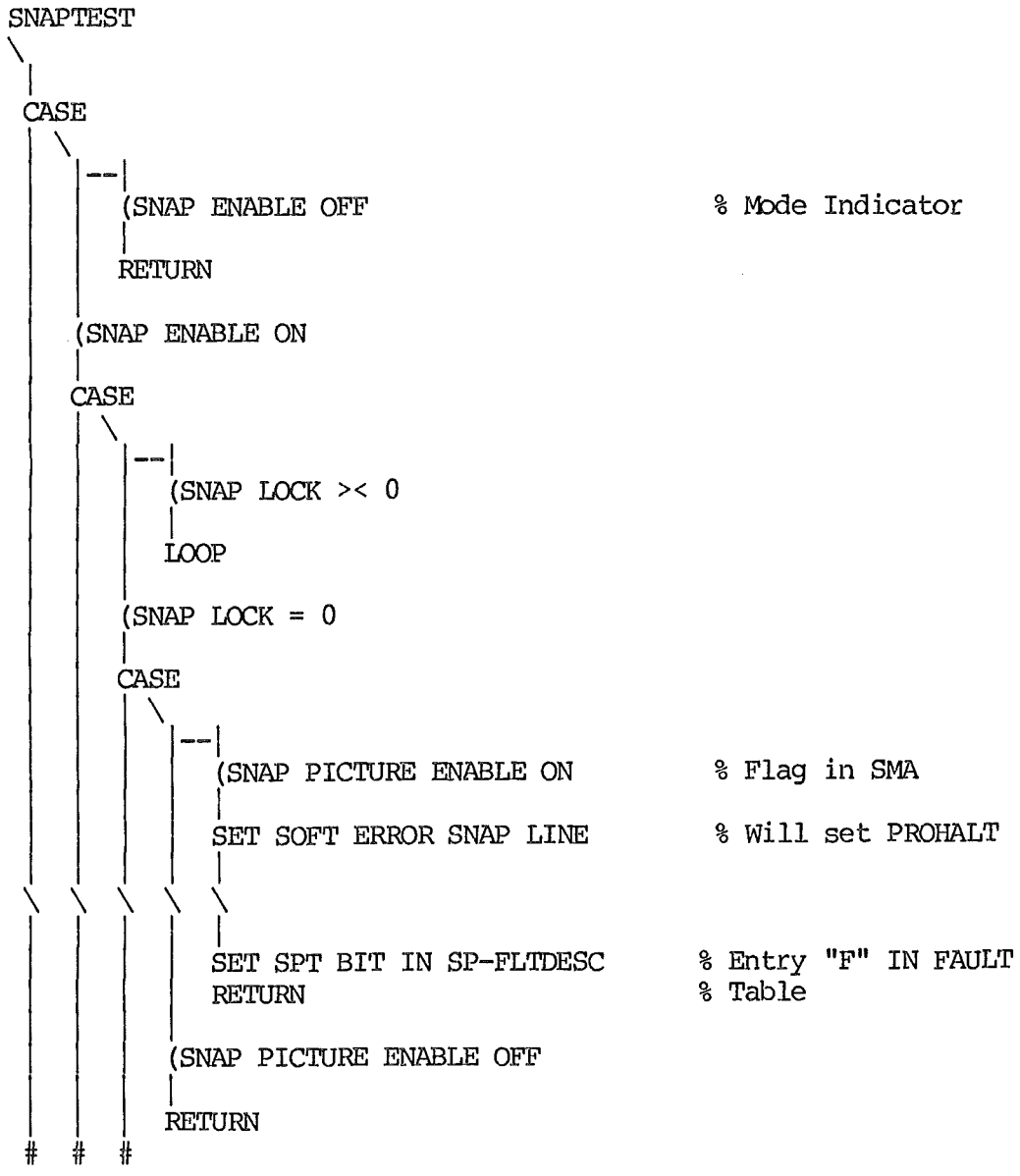
- when XM has determined that a SNAP PICTURE should be taken, it sets SOFT ERROR SNAP line and live freezes,
- MP shifts in the various chains and when it is finished with its own operations
- it wakes up XM firmware by forcing the microaddress '010F' Hex, which is one of the entries in the FAULT TABLE.

Note that SNAPTEST is performed as early as possible in HCP INTERFACE to avoid a significant alteration of the machine states.

It has to be mentionned also that when a MCACM fault exists (Address Error - Undigit Address or Limit Error), the MCACM ERROR ADDRESS and ERROR REPORT have to be read before anything else can be done - MCACM does not accept any command before these error registers are read. Since the read is destructive, the information is saved in MASS STORE (Address 97/98 Hex).

A flow of operation follows.

5.8.1 SNAP TEST FLOW



5.8.2 SNAP PICTURE REPORT

Refer to SDS called ' V500 SNAP '.

5.9 MEMORY ERROR REPORT

A Memory Error Report may be performed whenever MCACM reports either a Multiple-Bit Memory Board Failure or a Corrected Error.

In the case of a Multiple-Bit Memory Board Failure, the flow of operation is as follows:

- when XM sees the error, it dead freezes.
- MP wakes it up by forcing microaddress '010D'Hex, this allows XM to start serving the Uncorrectable Memory Parity Error Fault and build a Memory Error Report if required.

In the case of a Corrected Error, the ERROR Register in XM holds a description of the most recent error detected by MCACM. When a SST Op is executed, the MER STATUS, SMA [4324:2] and a MEMORY ERROR REPORT, address in SMA [4040:9], are updated if needed.

Note that a SYSTEM FLUSH resets the ERROR Register. To avoid losing possible Error Descriptors , the ERROR Register is saved locally in Mass Store (Address 'BF'Hex) before every SYSTEM FLUSH.

The Memory Error Report consists of four 10 digit words.

M.E.R. Word 1

MSD									LSD
T	EV0	LID-3	B-1	SW7	SW3	-	-	-	-
EV3	-	LID-2	B-0	SW6	SW2	-	-	-	-
EV2	-	LID-1	W-1	SW5	SW1	-	-	-	TA1
EV1	S	LID-0	W-0	SW4	SW0	-	-	-	TA0

T = Type of error,
0 = Non Fatal,
1 = Fatal.

EV3 -> EV0 = Error Vector (*)

S = Sub-System reporting
0 = Processor,
1 = I/O.

LID-3 -> LID-0 = Logical Card ID,

B-1/0 = Bank # (0-3)

W-1/0 = # of rightmost non zero syndrome word

SW7 -> SW0 = Rightmost non zero syndrome word

TA1/0 = Tag
00 = Fetch
01 = XM Read
10 = XM Write
11 = Invalid

(*) For values other than '0' or '1', only the following fields have a significant meaning: T, EVn,S and TAn. All other fields and words are not garanteed to be valid.

M.E.R. Word 2

MSD									LSD
-	MB3	S37	S33	S27	S23	S17	S13	S07	S03
-	MB2	S36	S32	S26	S22	S16	S12	S06	S02
-	MB1	S35	S31	S25	S21	S15	S11	S05	S01
-	MB0	S34	S30	S24	S20	S14	S10	S04	S00

MB3/0 = # of Memory Boards (0H means all 16 boards present)

S37 -> S30 = Word #3 Syndrome

S27 -> S20 = Word #2 Syndrome

S17 -> S10 = Word #1 Syndrome

S07 -> S00 = Word #0 Syndrome

M.E.R. Word 3

MSD									LSD
CTE-3	CT3	-	PID-3	-	MA-19	MA-15	MA-11	MA-07	MA-03
CTE-2	CT2	-	PID-2	-	MA-18	MA-14	MA-10	MA-06	MA-02
CTE-1	CT1	-	PID-1	MA-21	MA-17	MA-13	MA-09	MA-05	MA-01
CTE-0	CT0	PID-4	PID-0	MA-20	MA-16	MA-12	MA-08	MA-04	MA-00

CTE-3 -> CTE-0 = Reserved for Card Type Extension

CT3 -> CT0 = Card Type
CT3 set - Full Populated Board / 1 Megabit Chip
CT2 set - Half Populated Board / 1 Megabit Chip
CT1 set - Full Populated Board / 256Kbit Chip
CT0 set - Half Populated Board / 256Kbit Chip

PID-4 -> PID-0 = Physical ID Card (0 through F Hex)

MA-21 -> MA-00 = Block (40 digits) Mapped Address - binary

M.E.R. Word 4

MSD									LSD
-	-	-	-	UA-23	UA-19	UA-15	UA-11	UA-07	UA-03
-	-	-	-	UA-22	UA-18	UA-14	UA-10	UA-06	UA-02
-	-	-	-	UA-21	UA-17	UA-13	UA-09	UA-05	UA-01
-	-	-	UA-24	UA-20	UA-16	UA-12	UA-08	UA-04	UA-00

UA-24 -> UA-00 = Block (40 digits) Unmapped Address - binary

5.10 I/O HANDLING

The V500 I/O Handling is very similar to the one of the V300.

An I/O operation consists of the sequence CIO --> IIO --> IOC.

The I/O Scratchpad is located in the SHARED MEMORY AREA [5000:5000].

To each channel is assigned a 40 digit entry which is described as follows :

Current Buffer Begin Address	00 - 09
Current Buffer End Address	10 - 19
Extended Result Descriptor	20 - 31
Channel Busy	32 - 33
IOP Use	34 - 39

The number of channels has been expanded. Channels 80 to 87 and 90 to 97 are valid and dedicated to the MAINTENANCE PROCESSOR(s).

The address of the channel scratchpad is calculated by the formula : 5000 + (40 * Channel#) + Base of SMA

The I/O Mailbox (Channel 08) is located at 9600 + Base of SMA and is described as follows:

```
Address 9600 Base D: 000000BU00 ; BU   = Channel #
   "    9610    "  : 0000PP000I ; PP   = IOP OP
   "    9620    "  : 0VVVAAAA00 ; IVVV = Variant
   "    9630    "  : 0000AAAAAA ; A..A = Resolved Address
   "    9640    "  : 00BBBBBBBB ; B..B = Resolved Address
   "    9650    "  : BB0000LLLL ; L..L = B - A
   "    9660    "  : LLLLCCCCCC ; C..C = Resolved Address
   "    9670    "  : CCDDDDDDDD ; D..D = Resolved Address
   "    9680    "  : FF00000000 ; FF   = Mail Box Busy
```

5.11 TIMEOUT DETECTION

As was mentionned in section 4.9, special care has been taken to detect by firmware all the possible instances of Timeout.

A list of all the Ops and Pseudo Ops that have such a detection follows.

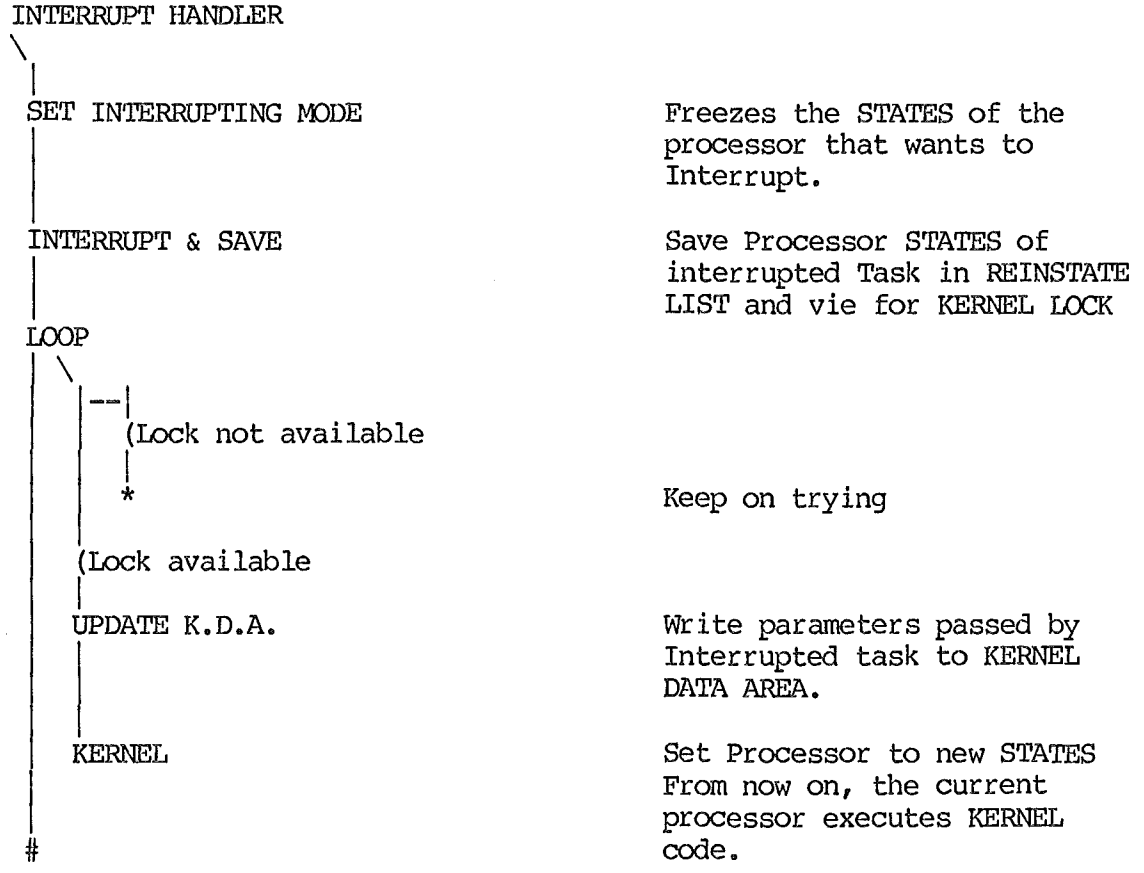
SLL - 37
SLD - 38
SEA - 39
SLT - 64
STB - 66
RET - 63
BRV - 93
IIO - 94
CIO - 85
ATE - 86
LOK - 60
ILS - 69
LDMT (Pseudo Op LOADMAT)
1E (Timeout detected by Fetch)

5.12 MULTIPROCESSOR CONSIDERATIONS

The V500 system can be made of up to 4 processors. To allow correct and easy communication between them, a SHARED MEMORY AREA (SMA) has been created (see section 5.3.4).

Some resources such as REINSTATE LIST POINTER, SNAP PICTURE POINTER, MEMORY AREA STATUS TABLE POINTER, MEMORY ERROR REPORT POINTER and KERNEL code are unique and shared by all the processors and their usage is protected by a Hardware Lock mechanism that will assure that only one processor at a time can use them.

An example describing it follows. It shows the sequence of operations performed for entering the KERNEL.



5.13 INITIALIZATION

When a processor is powered up, a series of states, registers and memory arrays have to be initialized before XM can be operational. The following list describes it.

MODULE	LOGIC FUNCTION	NAME	PATTERN (DGT)
XM	Scratch Pad	SP-0 to SP-F	0000000000
	Mass Store Address Reg	MSADREG	B3
	Mass Store	Address 99(MP date)	yyyymmdd00
		9A	0099999999
		9B(MP time)	nnnnnnnnn
		A8	0000999999
		A9 to AB	0000000000
		AC	0009990000
		AD	1009990000
		AE to BC	0000000000
		BD	0000000TTN
		BE to EF	0000000000
		F0	0000000000
		F1	0009990000
		F2	1000000000
		F3	1009990000
		F4	2000000000
		F5	2009990000
		F6	3000000000
		F7	3009990000
		F8	4000000000
		F9	4009990000
		FA	5000000000
		FB	5009990000
		FC	6000000000
		FD	6009990000
		FE	7000000000
		FF	7009990000
	Task Timer		99999999
	Interrupt Array Reg.	Mode Indicator	E4
		Miscellaneous Reg	02

MODULE	LOGIC FUNCTION	NAME	PATTERN (DGT)
MCACM	Base/Limit Table	Base 0	000000000
		Limit 0	009990000
		.	.
		.	.
		.	.
		Base 7	000000000
		Limit 7	009990000
		Base 8	000000000
		Limit 8	000000000
		.	.
		.	.
		.	.
		Base C	000000000
		Limit C	000000000
		Base D	Mem Limit - 10 K
		Limit D	Mem Limit
		Base E	000000000
		Limit E	000000000
		Base F	000000000
		Limit F	Mem Limit - 10 K
MEMORY	SHARED MEMORY AREA	Area Proc # 0	
		- Mobile Index Reg.	[0000:40] = 0
		- Mode Descriptor	[40:1] = 7
		- Miscellaneous	[41:959] = 0
		Area Proc # 1	[1000:1000]=0
		Area Proc # 2	[2000:1000]=0
		Area Proc # 3	[3000:1000]=0
		Common Area	
		- Miscellaneous	[4000:100] =0
		- Time of Day	[4100:10]=0YYYYMMDD0
		- System ID (*)	[4112:200]
		- Miscellaneous	[4312:16] =0

(*) System ID - All informations are in EBCDIC

Processor Type	(4112:20) = V500	- left justified
Specification Level	(4132:20) = TBD	
Shared System #	(4152:4) = F1,F2,F3 or F4	- right justified
Multiple Proc. #	(4156:4) = F1,F2,F3 or F4	- "
Serial #	(4160:20) = TBD	"
Memory Size	(4180:32) = nn	"
# of Central Proc.	(4212:4) = 0,1,2 or 3	- left justified
# of I/O Proc.	(4216:4) = TBD	
Firmware Level	(4220:88) = nn	"
Reserved	(4308:4) = 0000	

6. XM MICROCODE IDIOSYNCRACIES

Because of pecularities in the XM Hardware, some very strict rules had to be enforced in the design of its microcode.

6.1 DATA MOVEMENT

There exist some restrictions in the connections showed in the ICU matrix. (Refer to V500 Execute Module Specification - 1993 5204)

6.1.1 CONTROL OUT

Among other 'Control Card' sources, 'Control Out' is used to source the Interrupt Array Registers. They are Interrupt Register A, Interrupt Register B, Interrupt Mask A, Mode Register, Test Condition Register and Miscellaneous Register. They are 'slow sources': manipulation or test AND store of the result in the same clock is not allowed.

'Control Out' uses the least significant 10 bits (rightmost) of the 40 bit Data word.

6.1.2 INTERRUPT ARRAY REGISTERS

The Interrupt Array Registers are loaded only from DREG. A 2 clock operation is needed: clock 1, load DREG with right justified data (2 digits), clock 2, load specific register with data from DREG. Note that the most significant 8 digits in DREG have to be zero.

The handling of 'Interrupt Register A' is quite complicated because the register is instrumental in the IO Interrupt reporting. The following sequence must be performed to avoid any problem. The write into Interrupt Register A requires first to set its associated Mask Register to 0. Wait one clock before using Interrupt Register A (Read or Write). The following clock may then be used to restore the Mask register to its original value.

6.1.3 TASK TIMER

The Task Timer is an 8 digit counter which is read from the least significant 8 digits and written to the most significant 8 digits of the data word. It is loaded only from DREG and the least significant 2 digits have to be zero.

6.2 OPERAND QUEUE

When OF pre-reads data for XM, the length it uses is Mod 10. This means that if the length is 10 or less, the whole operand is in the OPERAND Queue. In all other cases the data read is not used but it HAS to be popped from the queue before OPLIT BRANCH.

Apart from the OF pre-read, data requested by XM may also come back through the OPERAND queue. This is done via CONDITIONAL READ (on DATA HIT) and UNCONDITIONAL READ memory commands. It is important to know that if a DATA HIT condition is detected, the OPERAND Queue A AND B are BOTH invalidated , even if only one operand has been pre-read. This means that to safely issue UNCONDITIONAL READ(S), the microcode has to issue CONDITIONAL READS for both A and B and pop the queue before.

6.3 READ BUS QUEUE

The status of the RBQ is described at any time by two flags - RBQ EMPTY and RBQ FULL.

In actuality, RBQ EMPTY means 'the queue will be empty next time a pop from it is issued' whereas RBQ FULL means ' the queue will be full next time a read is issued'.

This features allows the microcode to issue reads and/or pops in a one clock tight loop.

6.4 MEMORY REQUESTOR

6.4.1 LENGTH STATUS

The Memory Requestor Module is able to handle 3 different memory access streams - on 'A','B' or 'C'. This feature enables the microcode to offload the pointers handling into the hardware. Unfortunately its dynamic operation is limited by the fact that there exists only ONE length status for the three possible different memory accesses. This means that switching between 2 streams requires a one clock selection just to get the length status of the new stream.

6.4.2 WRITE OPERATIONS

There is a variety of WRITE commands, they are needed to fit the various situations that can occur.

'WRITE+' is used in the middle of a stream,

'WRITEL' is used at the end of a stream,

'WRITEOP' is used at the end of a stream when a 'pop lock unit' is needed,

'WRITE' is used when the stream length is 10 or less.

6.5 SYSTEM FLUSH

System Flush causes a few unwanted side effects that are corrected by microcode.

When issued, the flush causes the ERROR register to be reset; to avoid losing its content, it is saved in Mass Store (Address BF Hex).

The Memory requestor has a one deep command queue. To make sure that the FLUSH command effectively goes out when it is requested, it is followed by a 'fake command' - SELECT. The OPLIT BRANCH occurs no sooner than 3 additional clocks later. The delay allows enough time for the propagation of the FLUSH command to all other modules.

For hardware considerations, the CONTROL OUT SELECT bit is forced to be set from the time of the FLUSH until the time of OPLIT BRANCH.

6.6 FAULT HANDLER INTERFACE

OWhen a Synchronous Fault is detected during the execution of a given Op, the microcode for that instruction has to make sure that the Operand Queue related to it is popped before branching to the FAULT TABLE. There is no precaution to be taken as far as the READ BUS QUEUE is concerned. If the Fault requires an extension byte - ADDRESS ERROR or INVALID INSTRUCTION cases - it is passed in SP-14 (right justified).

Note that everytime an ENVIRONMENT CHANGE occurs, the microcode makes sure that there is no outstanding write error related to a previous operation by issueing a 'WRITE PC, READ PC, CHECK FOR MCACM FAULT' sequence before going on performing the LOADMAT itself. This is needed for reporting the correct source of the FAULT at all time.

6.7 OPLIT BRANCH

OPLIT BRANCH triggers many actions in the hardware, a few microcode requirements go along with them.

The Accumulator is located in Mass Store, it is qualified as a 'slow source'. To save one clock in the execution of any op involving the accumulator, it is required to set up the Mass Store Address Register to 'B3 Hex' - location of the Sign Exponent, Exponent, Sign Mantissa part - before OPLIT BRANCH time.

There is no information or state passing between Op executions except for the BRANCH Ops. Indeed, since an asynchronous interrupt/fault is only seen at OPLIT BRANCH time, the BRANCH Ops have tell to the interrupt/fault handler what is the Next Instruction Address (NIA), to be written into the interrupt/fault frame. This is done via F/F1. When set F/F1 means that NIA is found in the current Fetch Page - ASYL. For all other Ops, F/F1 and 0 have to be reset.

It is recommended not to execute any operation leading to possible LIVE FREEZE situation or writing into Mass Store at OPLIT BRANCH.

APPENDIX A - XM MICROCODE SYNTAX

0.0 SYMBOL RULES: (- @ > < + * ~ & <-)

Note: source indicates the SOURCE of a ICU connection.
dd stands for the two digit HEX address of a literal.

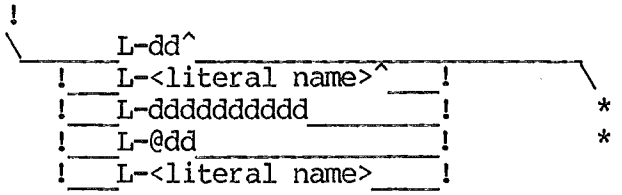
source-	AU A PORT input
-source	AU B PORT input
source>	LU A PORT input
<source	LU B PORT input
source@	PMUX A PORT input
@source	PMUX B PORT input
<-source	PTEST input
*source	PMREG input
+source	DREG input
&source	MS ADDRESS input
~source	MS DATA input
L-dd^	BYTE LITERAL DEFINITION
MRA-source	MEM REQUEST ADDRESS input
MRL-source	MEM REQUEST LENGTH input
WQ-source	MEM request DATA input
MEASREG-source	MEASUREMENT REGISTER input

0.1 STORE ELEMENTS LOADING:

AUREG<-	LOAD AU REGISTER
DREG<-	LOAD D REGISTER
PTREG<-	LOAD PTEST REGISTER
MSADR<-	LOAD MS ADDRESS REGISTER
MSIN<-	LOAD MS CONTENT
PMREG<-	LOAD DATA DEPENDENT PMUX REGISTER
SP-<source>-SPADRn<-	LOAD SCRATCH PAD
SP-<source>-<name><-	LOAD SCRATCH PAD

1. LITERAL: <LIT> defines a entry in the LITERAL FILE. The construct indicates either the CONTENT or the ADDRESS.
   The "^" symbol indicates the literal is a BYTE literal.
   The "@" symbol indicates the construct reflects the address of a literal.
   Symbolic names are allowed for literals. The actual ADDRESS and CONTENT of a symbolic literal should be put in the COMMENT field. Address and content are expressed in HEX form.

```
<LIT>                    --><byte literal> | <word literal>
<byte literal>           --> L-<byte literal content>^ |
                             L-<literal name>^
<byte literal content>   --> <literal address>
<word literal>           --> L-<word literal content>  |
                             L-@<literal address>      |
                             L-<literal name>
<word literal content>   --> 10-digit HEX value of the
                             content of literal
<literal address>        --> 2 digit HEX value of the
                             address of literal
<literal name>           --> SYMBOLIC name of literal
```



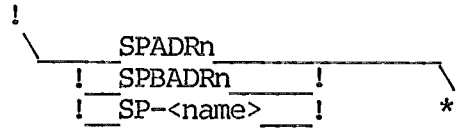
EXAMPLE:
- WORD LITERAL C000000003: L-C000000003
- WORD LITERAL AT 3A : L-@3A
- WORD LITERAL : L-POSITIVE
- BYTE LITERAL 35H : L-35^
- BYTE LITERAL : L-NEGATIVE^

NOTE: If word literal has leading zeros, the leading zeros are not necessary to appear in the construct.
EXAMPLE: WORD LITERAL 00000FFFFF: L-FFFFF

2. SP LOCATION <SPAadr> <SPBadr>
   ---------- SP LOCATION defines a location in Scratch Pad.
   <SPAadr> defines a location accessed from Port A and
   <SPBadr> defines a location accessed from PORT B.
   The actual address of a symbolic address should be
   put in the COMMENT field. The address is expressed
   in HEX form.

Note: refer to PART III for <primitive> constructs.

```
<SPAadr>           --> <primitive SPAadr> | <symbolic SPAadr>
<SPBadr>           --> <primitive SPBadr> | <symbolic SPBadr>
<primitive SPAadr> --> SPADR<SP address>
<symbolic  SPAadr> --> SP-<symbolic name>
<primitive SPBadr> --> SPBADR<SP address>
<symbolic  SPBadr> --> SP-<symbolic name>
<SP address>       --> 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
<symbolic name>    --> symbolic name of SP location
```

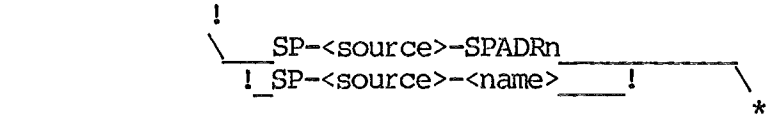


EXAMPLE:
   PRIMITIVE SPADR: SPADR0, SPADR5, SPBADR0
   SYMBOLIC SPADR: SP-DATA, SP-SOURCE

3. SP INPUT <SP INPUT>

SP INPUT defines a write operation to the Scratch Pad. Two parameters are needed: The SOURCE which feeds the SP and the destination LOCATION in SP to write. The actual address of a symbolic location should be put in the COMMENT field.

```
<SP INPUT>           --> <primitive SP input>|<symbolic SP input>
<primitive SP input> --> SP-<source module>-SPADR<SP address>
<symbolic  SP input> --> SP-<source module>-<NAME>
<source module>      --> SPA, FPA, RBQ, OPQA, MS, LU, LPC, CNC
<SP address>         --> 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
<name>               --> symbolic name of SP location
```



EXAMPLE:
   PRIMITIVE SP INPUT: SP-CNC-SPADR2, SP-LU-SPADR5
   SYMBOLIC SP INPUT: SP-CNC-SOURCE, SP-CNC-DATA

4. FPA ADDR <FPAadr>

FPA ADDR defines a location in the FETCH PAGE 1. The layout of FETCH PAGE is as follows:

0:	OP	Syllable
1:	A	Syllable
2:	B	Syllable
3:	C	Syllable
4:	PC	Syllable
5:	LENGTH	Syllable
6:	SPECIAL	Syllable
7:	SCRATCH	Syllable

<FPAADR> --> FOPSYL, FASYL, FBSYL, FCSYL, FPCSYL, FLENSYL, FSPESYL, FSCRSYL

5. FPB ADR <FPBadr>

   <FPBADR> --> FPB

6. CONTROL OUTPUT <CONTROL>
   -------------- CONTROL defines a entry accessed from the control output.

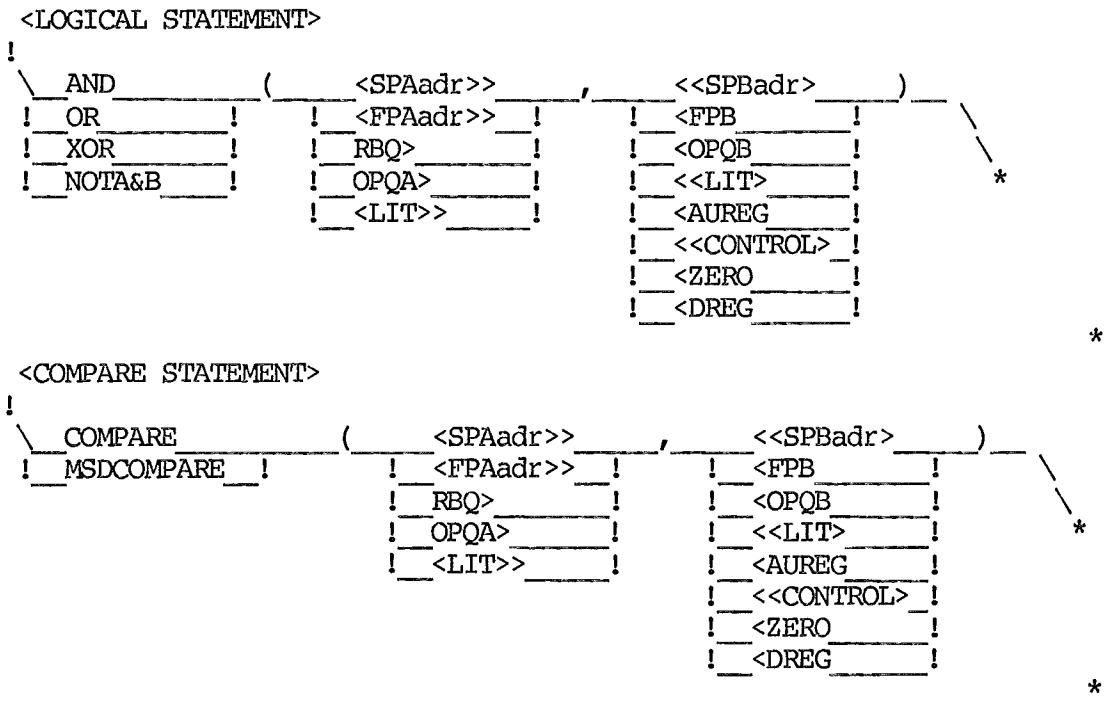
   <CONTROL> --> A-B, ERROR, ALEN, BLEN, CLEN, INT

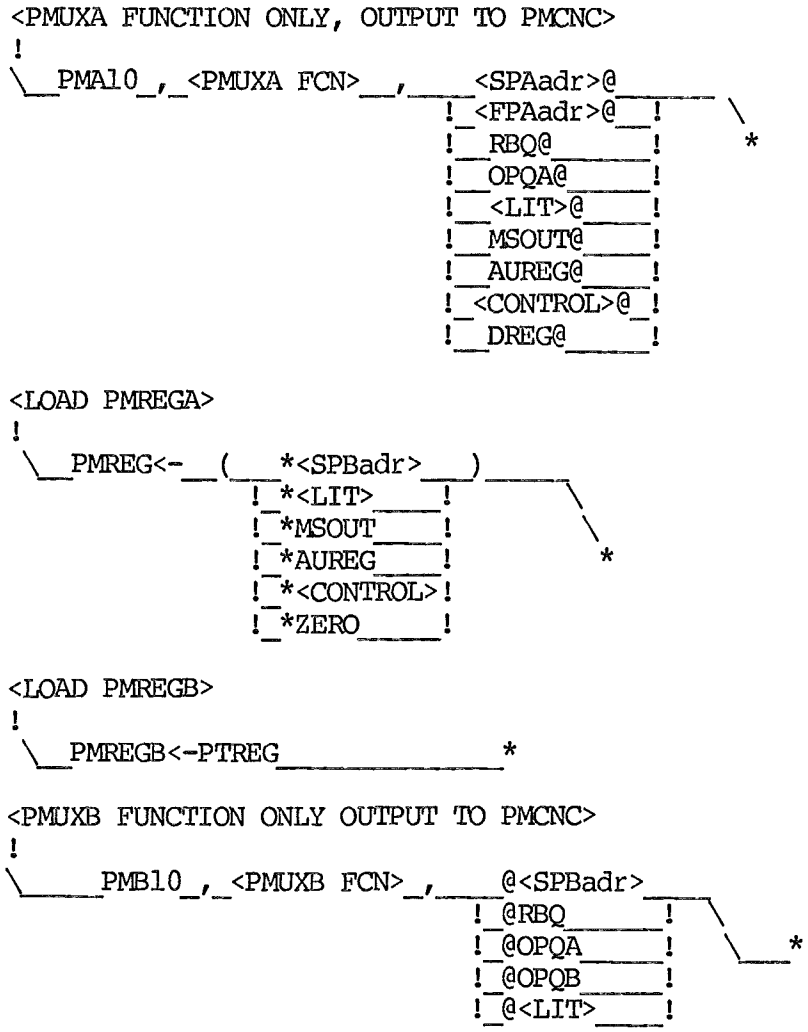
STATEMENTS: 1. All labels start at column 1.
---------- 2. All microcode statements start at column 8.
3. All comments start after column 50.

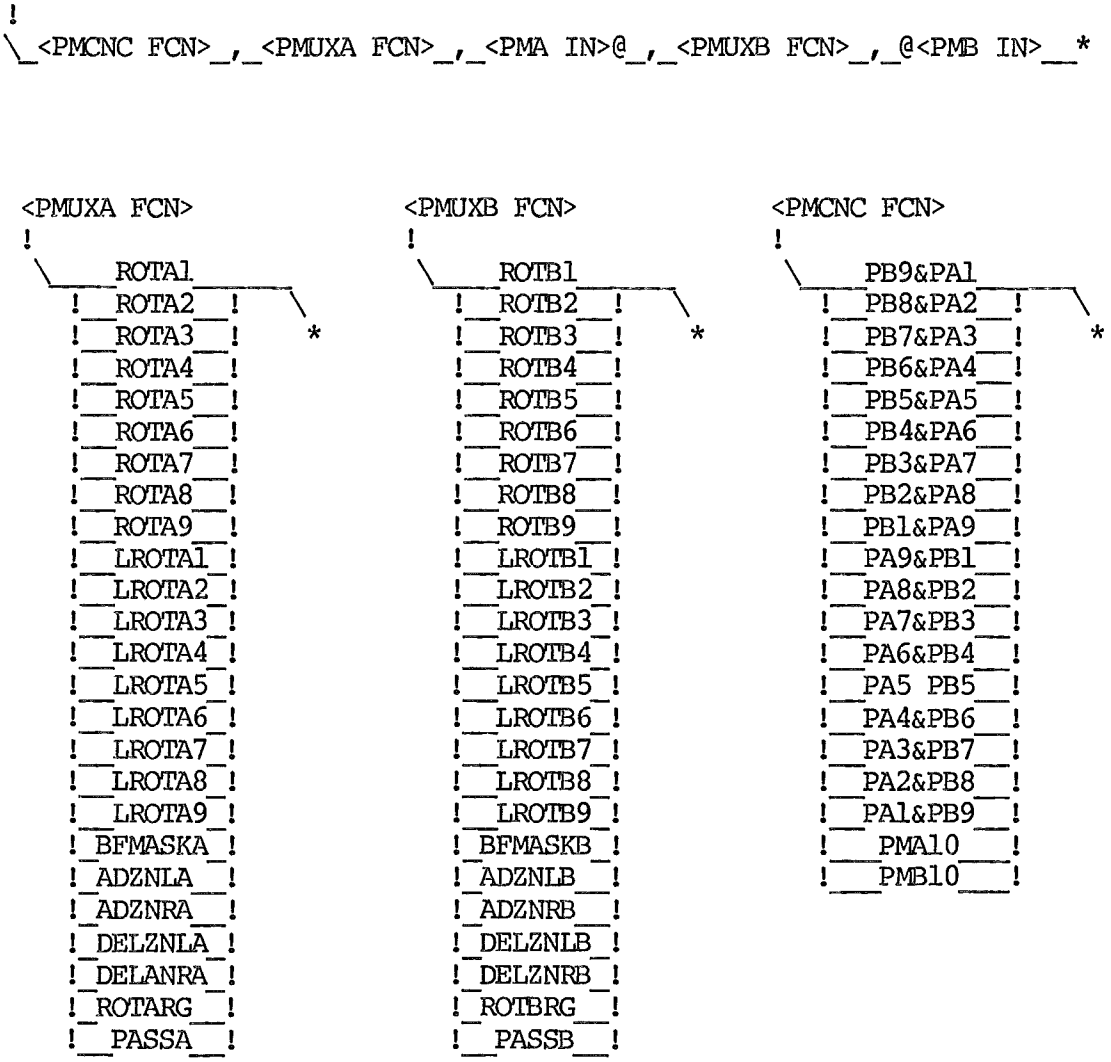
<ARITHMETIC STATEMENT>
!
__AUREG<-__(__<AUA SOURCE>___,___DADD_____,__<AUB SOURCE>__)__
!__BADD___! !
!__DSUB___! *
!__BSUB___!
!__DADD1__!
!__BADD1__!
!__DADDC__!
!__BADDC__!
!__DSUB1__!
!__BSUB1__!
!__DSUBOW_!
!__BSUBOW_!
!_DREVSUB_!
!_DREVSUB1_!

<AUA SOURCE>
!
_____<SPAadr>-_____
!___<FPAadr>-___! \
!___RBQ-_______! \
!___OPQA-______! *
!___<LIT>-_____!
!___MSOUT-_____!
!___AUREG-_____!
!___ZERO-______!
!___DREG-______!

<AUB SOURCE>
!
_____-<SPBadr>_____
!___-FPB_______! \
!___-OPQB______! \
!___-<LIT>_____! *
!___-MSOUT_____!
!___-AUREG_____!
!___-<CONTROL>_!
!___-DREG______!

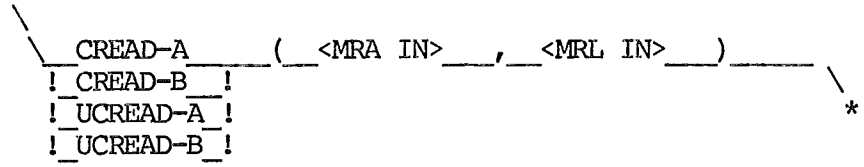

<LOGICAL STATEMENT>
AND
OR
XOR
NOTA&B
(
<SPAadr>>
<FPAadr>>
RBQ>
OPQA>
<LIT>>
,
<<SPBadr>
<FPB
<OPQB
<<LIT>
<AUREG
<<CONTROL>
<ZERO
<DREG
)
*
<COMPARE STATEMENT>
COMPARE
MSDCOMPARE
(
<SPAadr>>
<FPAadr>>
RBQ>
OPQA>
<LIT>>
,
<<SPBadr>
<FPB
<OPQB
<<LIT>
<AUREG
<<CONTROL>
<ZERO
<DREG
)
*


<PMUXA FUNCTION ONLY, OUTPUT TO PMCNC>
PMA10,<PMUXA FCN>,<SPAadr>@
<FPAadr>@
RBQ@
OPQA@
<LIT>@
MSOUT@
AUREG@
<CONTROL>@
DREG@
<LOAD PMREGA>
PMREG<-(*<SPBadr>)
*<LIT>
*MSOUT
*AUREG
*<CONTROL>
*ZERO
<LOAD PMREGB>
PMREGB<-PTREG
<PMUXB FUNCTION ONLY OUTPUT TO PMCNC>
PMB10,<PMUXB FCN>,@<SPBadr>
@RBQ
@OPQA
@OPQB
@<LIT>

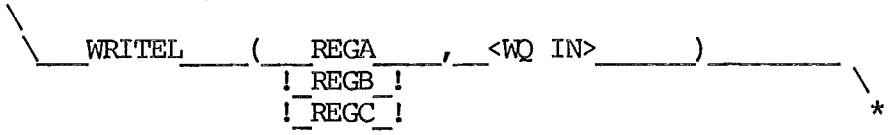

<CNC FCN, PMUXA FCN, MPUXB FCN, OUTPUT TO PMCNC >
<PMCNC FCN>,<PMUXA FCN>,<PMA IN>@,<PMUXB FCN>,@<PMB IN>*
<PMUXA FCN>
ROTA1
ROTA2
ROTA3
ROTA4
ROTA5
ROTA6
ROTA7
ROTA8
ROTA9
LROTA1
LROTA2
LROTA3
LROTA4
LROTA5
LROTA6
LROTA7
LROTA8
LROTA9
BFMASKA
ADZNLA
ADZNRA
DELZNLA
DELANRA
ROTARG
PASSA
<PMUXB FCN>
ROTB1
ROTB2
ROTB3
ROTB4
ROTB5
ROTB6
ROTB7
ROTB8
ROTB9
LROTB1
LROTB2
LROTB3
LROTB4
LROTB5
LROTB6
LROTB7
LROTB8
LROTB9
BFMASKB
ADZNLB
ADZNRB
DELZNLB
DELZNRB
ROTBRG
PASSB
<PMCNC FCN>
PB9&PA1
PB8&PA2
PB7&PA3
PB6&PA4
PB5&PA5
PB4&PA6
PB3&PA7
PB2&PA8
PB1&PA9
PA9&PB1
PA8&PB2
PA7&PB3
PA6&PB4
PA5 PB5
PA4&PB6
PA3&PB7
PA2&PB8
PA1&PB9
PMA10
PMB10

EXTERNAL COMMAND

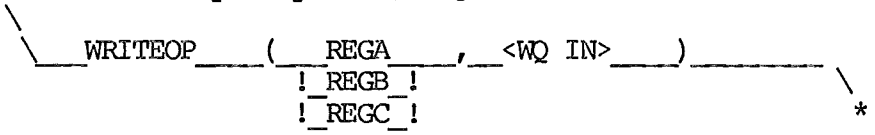
<CONDITIONAL READ & UNCONDITIONAL READ>



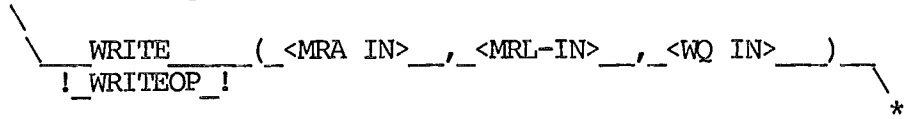
<write & complete the current write stream by reg>



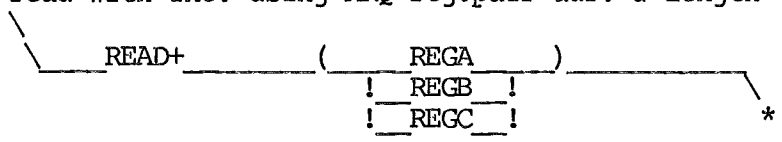
<write with op complete , signal lock unit>



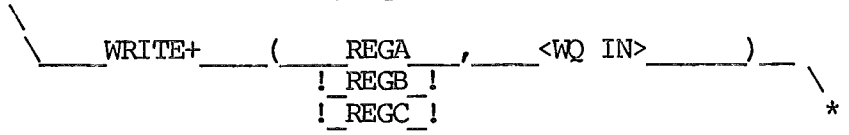
<write through without using registers>

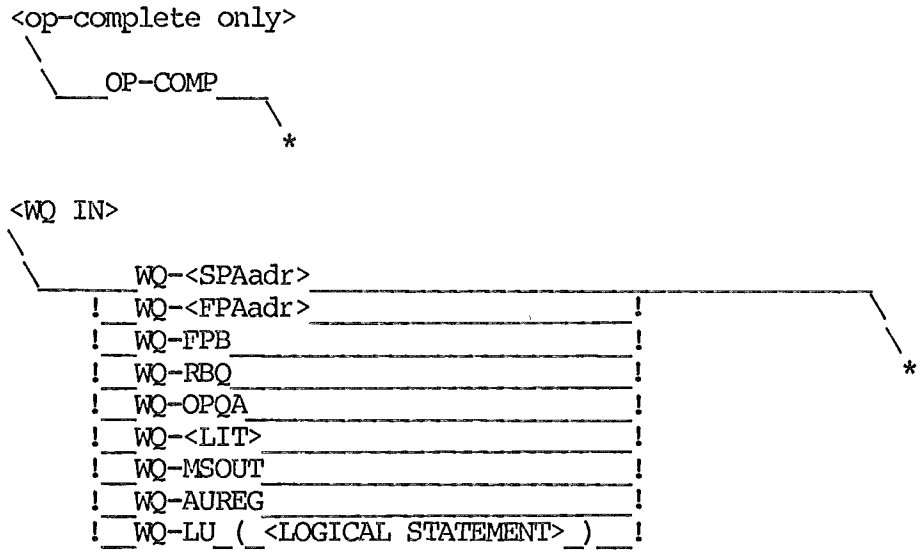


<read with inc. using MRQ reg.pair adr. & length>

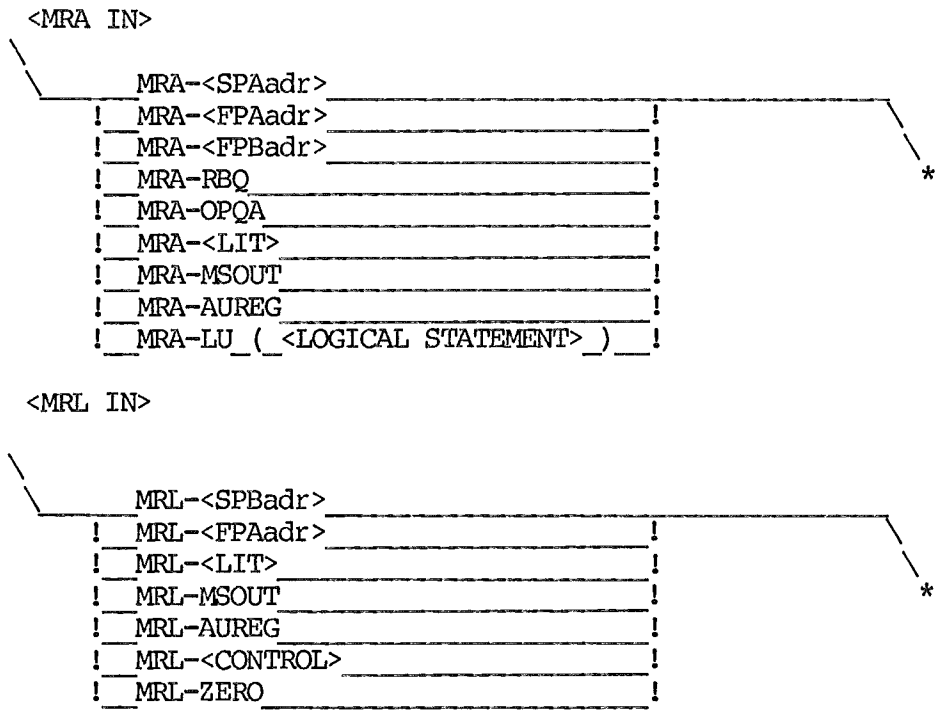


<write with inc. using rq reister pair adr & length>

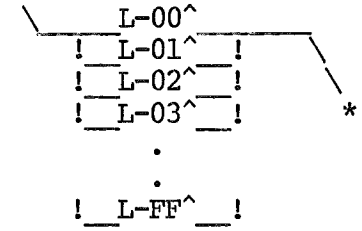



<op-complete only>
OP-COMP
*
<WQ IN>
WQ-<SPAadr>
WQ-<FPAadr>
WQ-FPB
WQ-RBQ
WQ-OPQA
WQ-<LIT>
WQ-MSOUT
WQ-AUREG
WQ-LU (<LOGICAL STATEMENT>)
*

Memory Requestor Address & length input syntax



<Byte literal syntax>

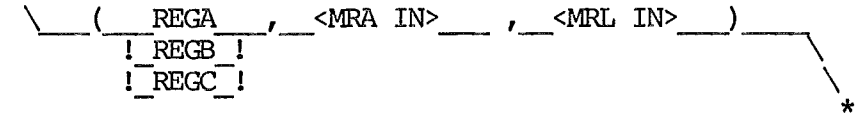


Note: Byte literal must be in hex format as CREG bits [92:8]

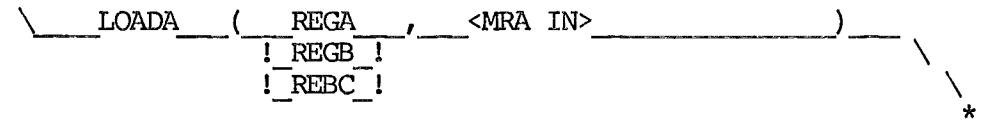
<different ICU Byte literal syntax>
Note: dd is a two digit HEX Literal Address

L-dd^-	Byte Literal to AUA	input
-L-dd^	Byte Literal to AUB	input
L-dd^>	Byte Literal to LUA	input
<L-dd^	Byte Literal to LUB	input
L-dd^@	Byte Literal to PMUX A	input
@L-dd^	Byte Literal to PMUX B	input
+L-dd^	Byte Literal to DREG	input
*L-dd^	Byte Literal to PMREGA	input
WQ-L-dd^	Byte Literal to WQ	input
MRL-L-dd^	Byte Literal to MRL	input
MRA-L-dd^	Byte Literal to MRA	input

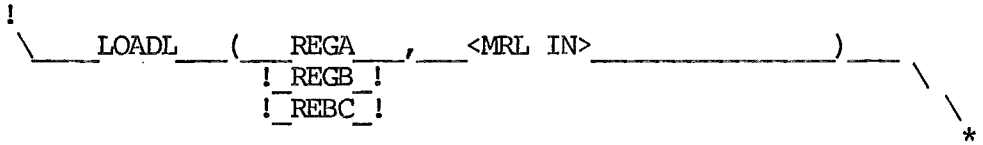
<load MRQ register pair A B or C command>

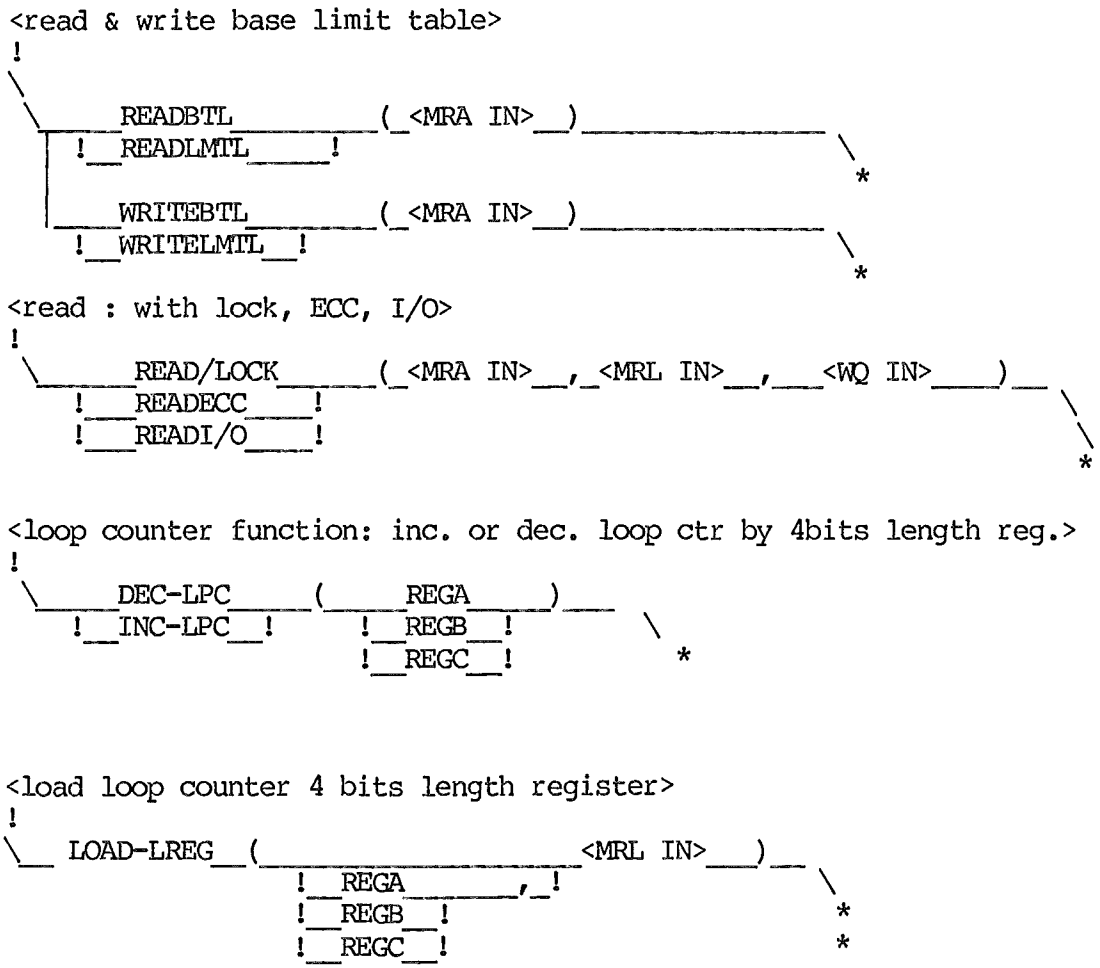


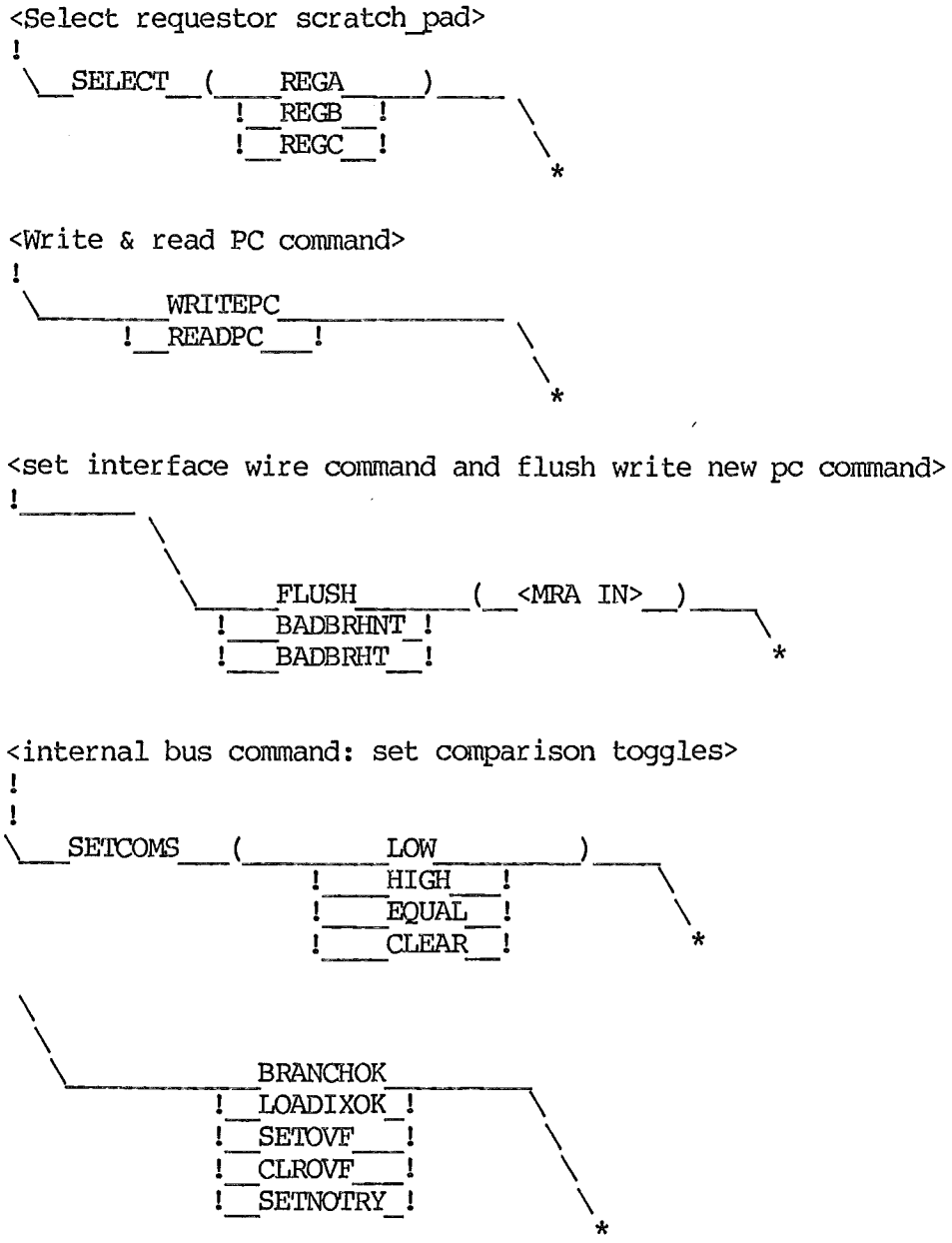
<Load address register only>

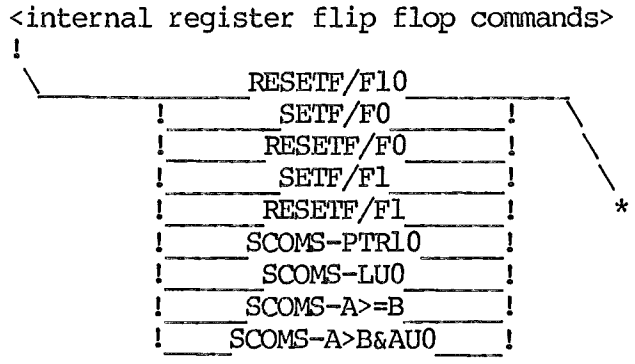


<Load length register only>

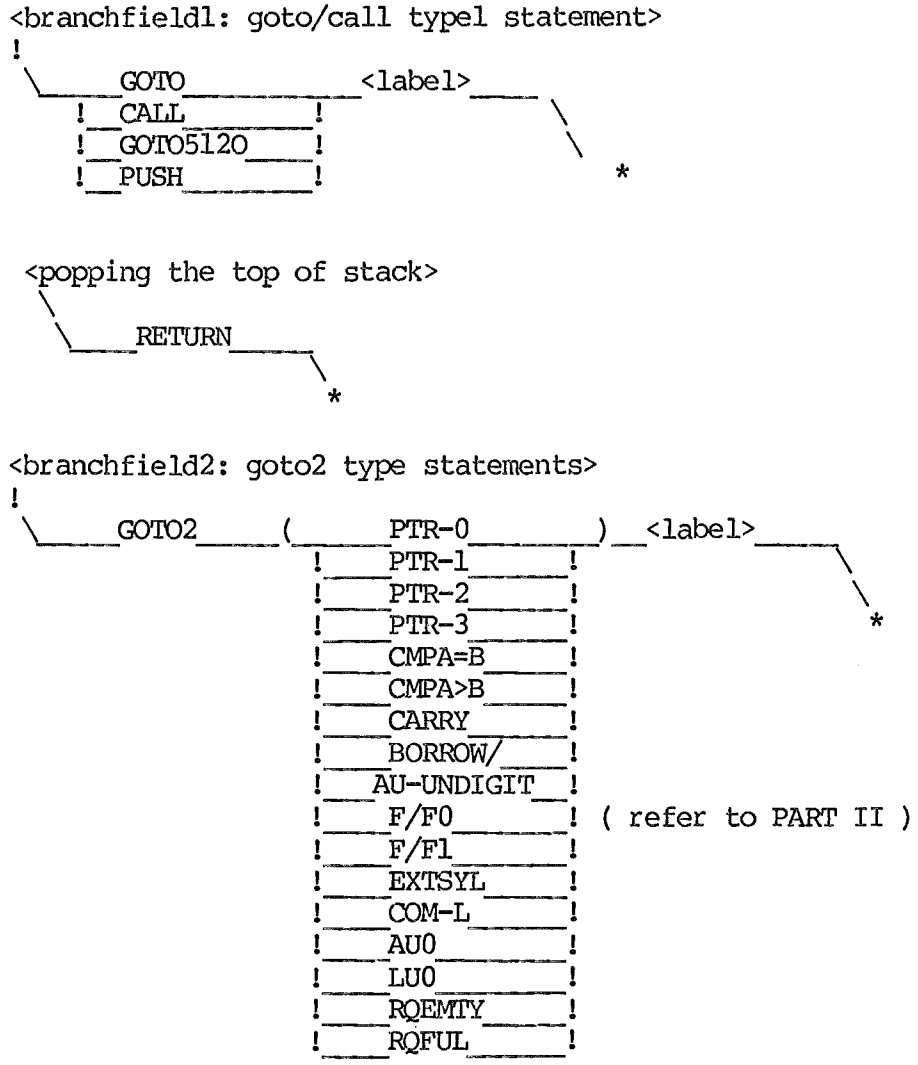


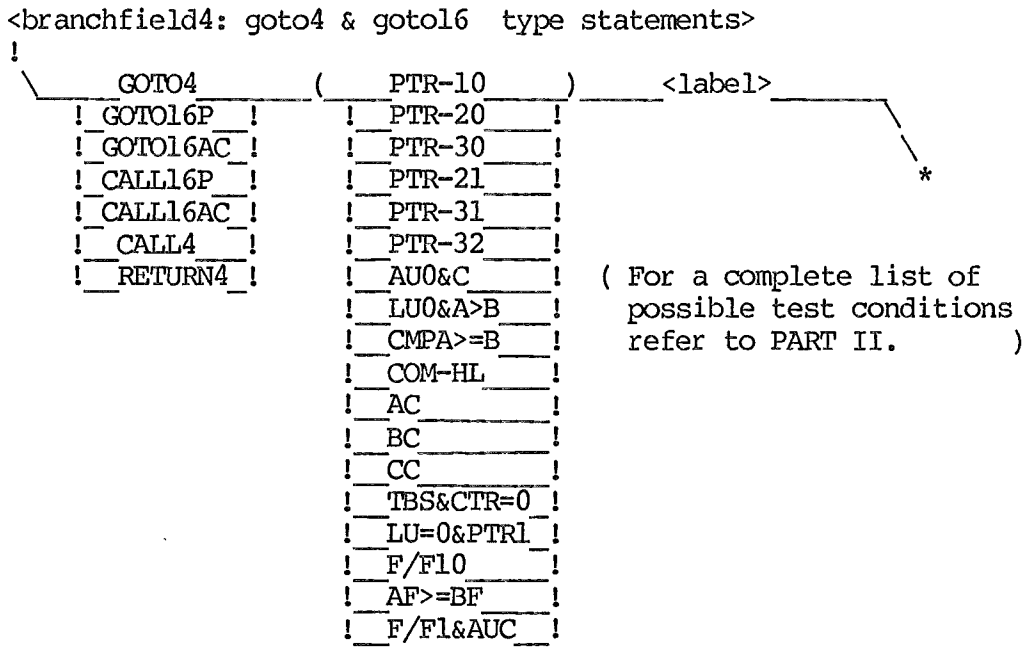

<read & write base limit table>
READBTL
<MRA IN>
READLMTL
WRITEBTL
<MRA IN>
WRITELMTL
<read : with lock, ECC, I/O>
READ/LOCK
<MRA IN>
<MRL IN>
<WQ IN>
READECC
READI/O
<loop counter function: inc. or dec. loop ctr by 4bits length reg.>
DEC-LPC
INC-LPC
REGA
REGB
REGC
<load loop counter 4 bits length register>
LOAD-LREG
<MRL IN>
REGA
REGB
REGC

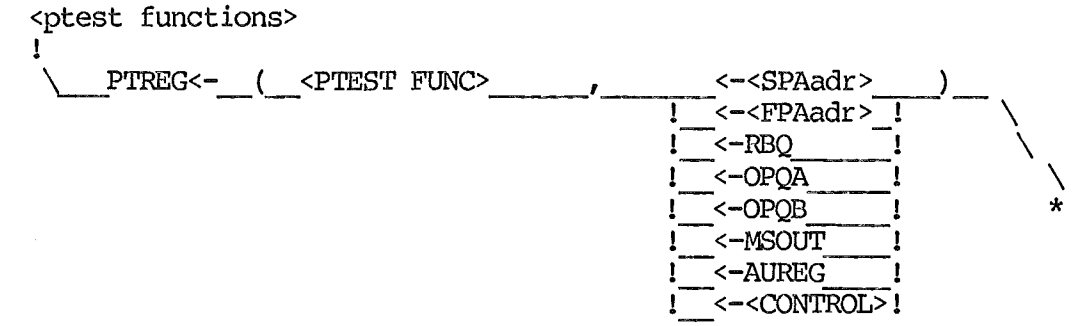

<Select requestor scratch_pad>
SELECT
REGA
REGB
REGC
<Write & read PC command>
WRITEPC
READPC
<set interface wire command and flush write new pc command>
FLUSH
<MRA IN>
BADBRHNT
BADBRHT
<internal bus command: set comparison toggles>
SETCOMS
LOW
HIGH
EQUAL
CLEAR
BRANCHOK
LOADIXOK
SETOVF
CLROVF
SETNOTRY



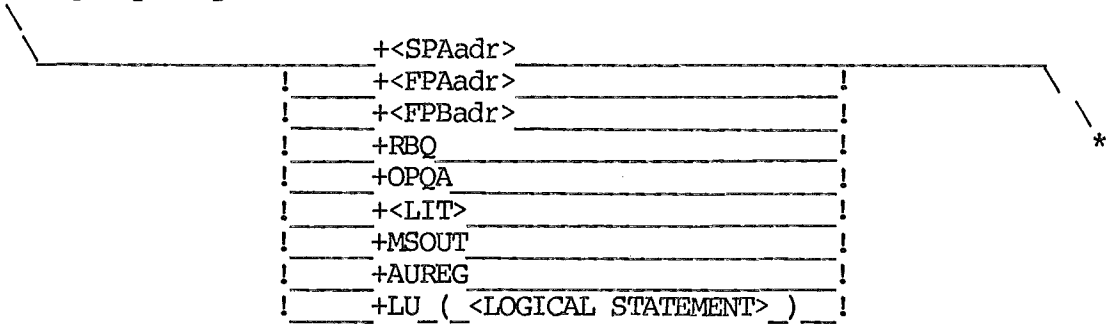
NOTE: For detail definitions, refer PART II.

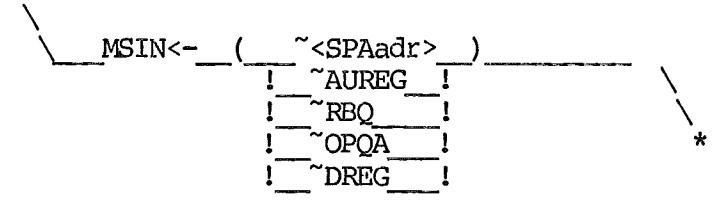

<branchfield1: goto/call type1 statement>
GOTO
<label>
CALL
GOTO5120
PUSH
*
<popping the top of stack>
RETURN
*
<branchfield2: goto2 type statements>
GOTO2
(
PTR-0
PTR-1
PTR-2
PTR-3
CMPA=B
CMPA>B
CARRY
BORROW/
AU-UNDIGIT
F/F0
F/F1
EXTSYL
COM-L
AU0
LU0
RQEMTY
RQFUL
) <label>
*
(refer to PART II)

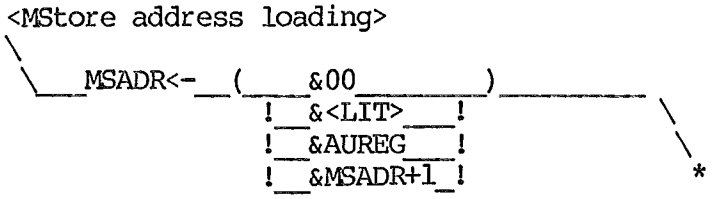

<branchfield4: goto4 & goto16 type statements>
GOTO4
GOTO16P
GOTO16AC
CALL16P
CALL16AC
CALL4
RETURN4
(
PTR-10
PTR-20
PTR-30
PTR-21
PTR-31
PTR-32
AU0&C
LU0&A>B
CMPA>=B
COM-HL
AC
BC
CC
TBS&CTR=0
LU=0&PTR1
F/F10
AF>=BF
F/F1&AUC
<label>
*
(For a complete list of
possible test conditions
refer to PART II.)



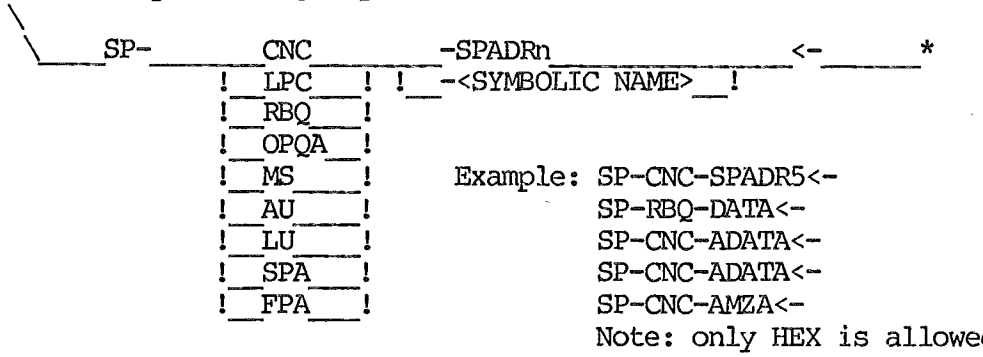
<ptest func>
No standard names are defined for PTEST functions.
Any meaningful name is allowed as long as the actual function code is put in the COMMENT field.





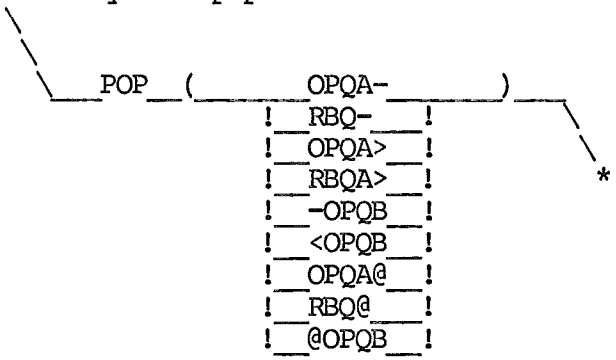


<Scratch pad storage syntax> === <SP INPUT><-



Example: SP-CNC-SPADR5<-
SP-RBQ-DATA<-
SP-CNC-ADATA<-
SP-CNC-ADATA<-
SP-CNC-AMZA<-
Note: only HEX is allowed for n.

<Data queues pop without store>



APPENDIX B - V500/V300 Code Space & Performance Comparison

In this comparison, the V300 figures are derived from APPENDIX B of the B4900 PROCESSOR Product Specification. Because of the complexity of both the instruction set and the hardware that executes it, estimates rather than accurate numbers are provided.

INSTRUCTION	CODE SPACE		PERFORMANCE V500/V300		
	V500	V300	MAX	AVE	MIN
INC/DEC/ADD/SUB/CPN	1060	1000			
Optimal			4		
General			5		2.5
MPY	700	1000			
Optimal			25	3	.5
General			2		
DIV	810	700			
Optimal			3		
General			3		
MVD/MVW/MVC	400	400	1.8		
MVR	130	130		5	
MVL	80	178		3	
MVA/MVN	890	1500			
Optimal			12	6	1
General			8	2.5	1
TRN	275	200	5	3	2
SCAN	450	1200	5.5	2.5	.8
BRANCH	175	210	5	3	1
HALT	60	76		1	
BOT/BZT	80	80	4	2	1.5
BST/BRT			4	2.5	1.5
AND/OR/NOT	350	415	4	2.5	1
CPA	280	350	4.5		
SLT/STB	250	236	2		1.5
SEA	240	358	9	3.5	.7
SLL/SLD	160		2	1.5	1.5
EDT	660	600	4.5	2	.7
LOK/ILS/MLS	270		3	1.5	
LIX/SIX	60		13	3	.8
WHR/SST	160			1	
ATE	400				
MOP	23	43			
D2B/B2D	510	490			

INSTRUCTION	CODE SPACE		PERFORMANCE V500/V300		
	V500	V300	MAX	AVE	MIN
IAD/IAS/ISU/ISS	62	97		1	
ILD/IST			3	2	
IMI	38	43		1.5	
IMU/IMS	160				
RLD/RST	55		2	1.5	1
RAA/RAS/RSU/RSS	390		4	1.5	1
RMU/RMS	365		3.5	2	1
ACM	175		2	1.5	1
RDV/RDS	210	210	5	2	.2
IOC/CIO/IIO	460	569		1.5	
SRD/RAD	100	132		1.4	
RDT/RST	26				
MVS/CPS/HSH	770		13	8	6
BRV	100	182		2	
NTR/EXT	64			2	
ASP	27	40		2.5	
VEN	130	130		1	
RET	95	160		2	
INT	20	44			
LDMT	350			1	
IP	180	150			
BCT	*	*		1.2	
HCL	*	*		1.2	
HCP	*	*		1.5	

* ECROUT/HCROUT/IPROUT/HCL/BCT/HCP = 1000
GROVER/FAULT/HHCL/INTERRUPT = 1050