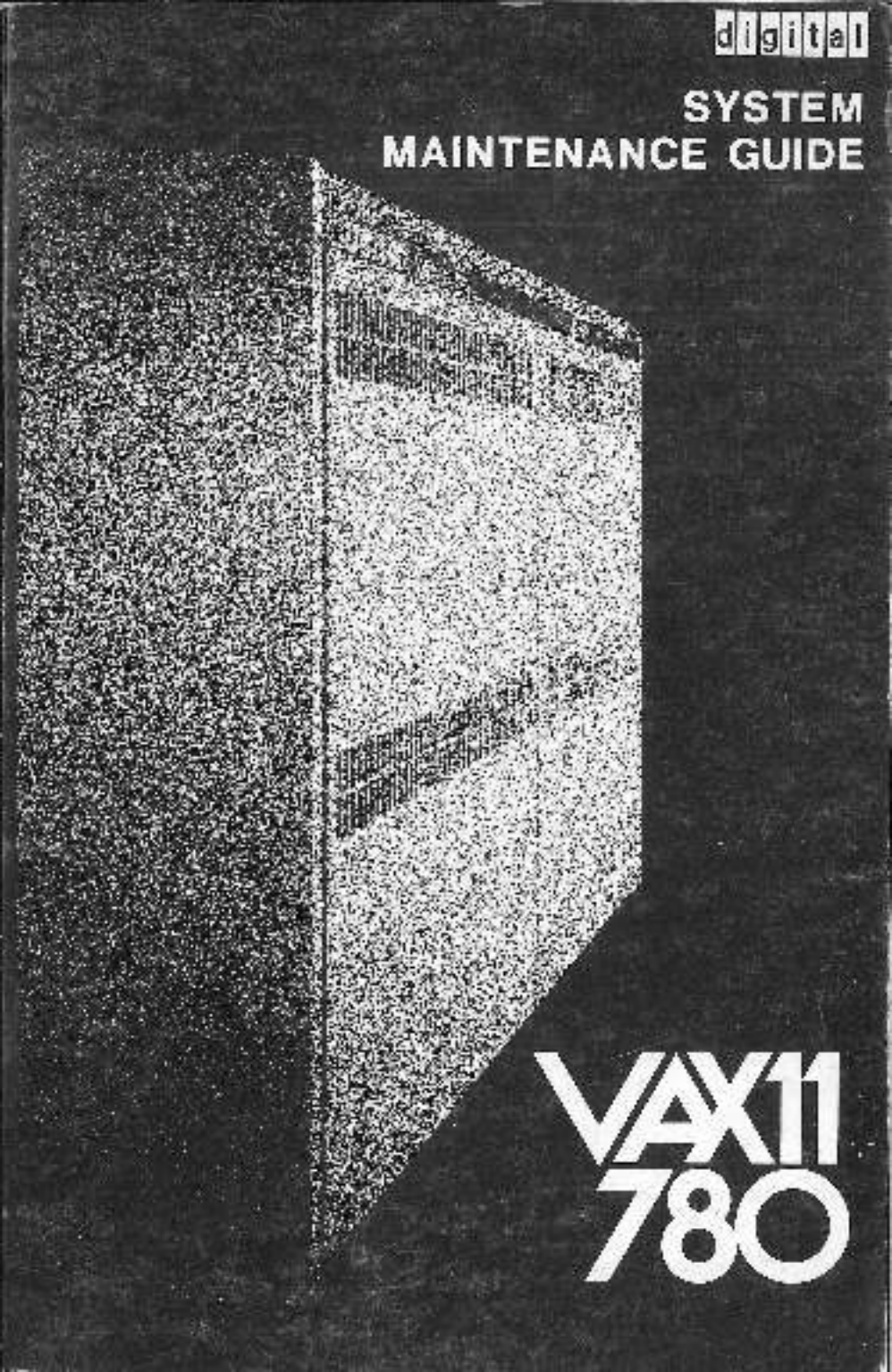
digital

SYSTEM MAINTENANCE GUIDE



VAX11 780

EK-11780-PG-001

VAX-11/780
SYSTEM MAINTENANCE GUIDE

digital

First Edition, December 1978

CONTENTS

SECTION 1 INTRODUCTION

SECTION 2 ARCHITECTURE

CONTENTS (Continued)

CONTENTS (Continued)

SECTION 4 CONFIGURATION/JUMPERS

CONTENTS (Continued)

CONTENTS (Continued)

SECTION 1
INTRODUCTION

INTRODUCTION

This handbook is designed as a single source summary of troubleshooting, maintenance, operating, and programming information on the VAX 11/780 computer system. The materials provided supplement the detailed information available in the hardware and software manual sets, the print sets, and program listings. The handbook will serve as a quick reference for Digital field service, manufacturing, training, and engineering personnel.

The materials included consist of tables, lists, listings, diagrams, and procedures. This format assumes that readers are familiar with the VAX-11/780 system and the nomenclature and procedures.

For explanations of these materials and for further details on the VAX-11/780 system, see the VAX-11/780 Microfiche Library and the items listed in the following three tables.

Hard copy manuals can be ordered from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532
Attn: Printing and Circulation Services (NR2/N15)
Customer Services Section

For information concerning microfiche libraries, contact:

Digital Equipment Corporation
Micropublishing Group
Cronin Drive
Bedford, MA

VAX-11/780 HARDWARE MANUALS

Document Title	Control Number	Kind
VAX 11/780 Power System Technical Description	EK-PS780-TD-001	Fiche
VAX 11/780 System Installation Manual	EK-[illegible]780-IN-001	Hard copy
DF780 Diagnostic System User's Guide	EK-DS780-UG-001	Hard copy
DS780 Diagnostic System Technical Description	EK-DS780-TD-001	Fiche
FP780 Floating Point Processor Technical Description	EK-FP780-TD-001	Fiche
RB730/UDA50 Subsystem Technical Documentation	EK-[illegible]-TD-001	Fiche
VAX 11/780 Central Processor Technical Description	EK-KA780-TD-001	Fiche
VAX-11/780 Memory System Technical Description	EK-MS780-TD-001	Fiche
TR780 Unibus Adapter Technical Description	EK-TR780-TD-001	Fiche
KA780 Console Interface Technical Description	EK-[illegible]780-TD-001	Fiche
VAX-11/780 Software Handbook	EB-[illegible]6	Hard copy
VAX-11/780 Architecture Handbook	EB-[illegible]6	Hard copy

VAX-11/780 PERIPHERAL MANUALS

Document Title	Control Number	Form
LA36/LA35 DECwriter II User's Manual	EK-LA36i-OP-003	Hard copy
VT-52 DECscope Maintenance Manual	EY-VT52-MM-001	Hard copy
RH70 Disk Subsystem User's Manual	EK-RH70-UG-001	Hard copy
RP05/RP06 DEC Disk Storage Drive Technical Manual	EK-2012 EK-RP1/RP2-TM	Hard copy
UNIBUS/LSI-11/PDP-11 DECtape Transport Maintenance Manual	EK-OTE10-TM-001	Hard copy
RX01/RX11 Floppy Disk Maintenance Manual	EK-RX01-MM-002	Hard copy
LSI-11, PDP-11/03 User's Manual	EK-LSI11-TM-001	Hard copy
LP11/LA11/LA11 Line Printer User's Manual	EK-LP11-OP-001	Hard copy
CR11/CM11 Card Reader System Manual	EK-CR11-TM-001	Hard copy
LA180 DEC Printer Maintenance Manual	EK-LA180-MM-002	Hard copy
PDP-11 Peripherals Handbook	EB-20986	Hard copy

VAX-11/780 SOFTWARE DOCUMENTATION

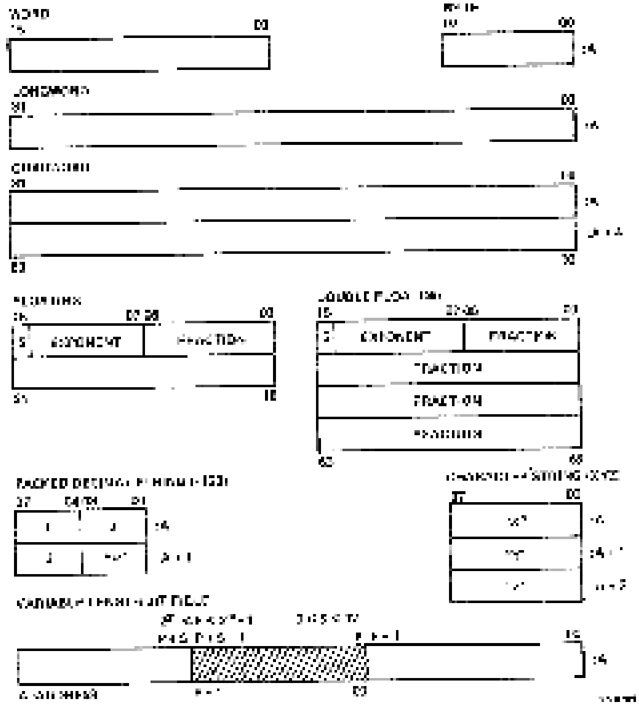
VOLUME	DOCUMENT TITLE	DEC ORDER NUMBER
VOLUME 1A System Reference	VAX/VMS Primer	AA-D030A-TE
	VAX/VMS Summary Description	AA-D032A-TE
	VAX/VMS Information Directory	AA-D033A-TE
	VAX/VMS Release Notes	AA-D035A-TE
	VAX-11 SoftLabs Installation Guide	AA-D071A-TE
	VAX/VMS System Services Reference Manual	AA-D034A-TE
VOLUME 1B System Reference	VAX/VMS Command Language User's Guide	AA-D031A-TE
	VAX-11 Linker Reference Manual	AA-D032A-TE
	VAX-11 Symbolic Debugger Reference Manual	AA-D036A-TE
VOLUME 1C System Reference	VAX-11/782 FM Programming Reference Manual	AA-D033A-TE
	VAX-11/780 FM User's Guide	AA-D120A-TE
	VAX-11 MACRO Language Reference Manual	AA-D037A-TE
	VAX-11 MACRO User's Guide	AA-D038A-TE
VOLUME 2A System Programmer	VAX-11 Common Run-Time Procedure Library Reference Manual	AA-D039A-TE
	VAX-11 Test Writing Reference Manual	AA-D040A-TE

VOLUME 2B System Procedures	VAX/VMS Operator's Guide	AA-D022A-TB
	VAX/VMS System Manager's Guide	AA-D023A-TB
	VAX/VMS System Messages and Recovery Procedures Manual	AA-D017A-TB
	VAX/VMS DBTP User's Guide	AA-D671A-TB
	VAX-11 Disk Save and Compress User's Guide	AA-D735A-TB
VOLUME 3 VAX/VMS I/O	VAX/VMS I/O User's Guide	AA-D029A-TB
	Introduction to VAX-11 Record Management Services	AA-D032A-TE
	VAX-11 Record Management Services Reference Manual	AA-D031A-TE
	VAX-11 Record Management Services User's Guide	AA-D781A-TE
VOLUME 4 DNS 11/ DDCMP	IAS/RSX-11M DDT-11 MACRO Programmer's Reference Manual	AA-D102A-TC
	Introduction to RMS-11	AA-H001A-TC
	RSX-11M FMS-11 Utilities User's Guide	AA-D083A-TC
	VAX-11 DDCMP Reference Manual	AA-D117A-TE
VOLUME 5A Options Software	VAX-11 FORTRAN IV-PLUS Language Reference Manual	AA-D034A-TE
FORTRAN IV PLUS	VAX-11 FORTRAN IV PLUS User's Guide	AA-D035A-TE

VOLUME 4B Optional Software	PDP-11 FORTRAN Language Reference Manual	DEC-11-LFRMA-C-D
FORTRAN IV	PDP-11 FORTRAN Language Reference Manual Update Notice No. 1	DEC-11-LRMUA-C-D51
	IAS-RSX-11 FORTRAN IV User's Guide	DEC-11-LUFUA-D-D
VOLUME 5C Optional Software	PDP-11 COBOL Language Reference Manual	AA-1743D-TC
	PDP-11 COBOL User's Guide	AA-1737C-TC
COBOL	PDP-11 COBOL Pocket Guide	AA-1743C-TC
VOLUME 5I	BASIC PLUS 2 Language Reference Manual	AA-0139A-TK
BASIC PLUS 2	BASIC PLUS-2 RSX 11X/IAS User's Guide	AA-0137A-TC
VOLUME 5B	DECnet-VAX System Manager's	AA-D502A-TK
Optional Software	Guide	
DECnet	DECnet-VAX User's Guide	AA-D501A-TK
VOLUME 5F Optional Software	User's Guide to DATATRIEVE-11	AA-0742A-TC
DATATRIEVE		

SECTION 2
ARCHITECTURE

DATA TYPE	SIZE	RANGE (DECIMAL)	
INTEGER		SIGNED	UNSIGNED
BYTE	8 BITS	-128 TO +127	0 TO 255
WORD	16 BITS	-32768 TO +32767	0 TO 65535
LONGWORD	32 BITS	-2^{31} TO $+2^{31}-1$	0 TO $2^{32}-1$
QUADWORD	64 BITS	-2^{63} TO $+2^{63}-1$	0 TO $2^{64}-1$
FLOATING POINT		$\pm 2.9 \times 10^{-37}$ TO 1.7×10^{38}	
FLOATING	32 BITS	APPROXIMATELY SEVEN DECIMAL DIGITS PRECISION	
DOUBLE FLOATING	64 BITS	APPROXIMATELY SIXTEEN DECIMAL DIGITS PRECISION	
PACKED DECIMAL STRING	0 TO 16 BYTES (31 DIGITS)	NUMERIC, TWO DIGITS PER BYTE SIGN IN LOW HALF OF LAST BYTE	
CHARACTER STRING	0 TO 65535 BYTES	ONE CHARACTER PER BYTE	
VARIABLE-LENGTH BIT FIELD	0 TO 32 BITS	DEPENDENT ON INTERPRETATION	



Mode	Register	Dec	Name	Assembler	B	W	A	D	Q	#0	#1	Instruction
0 0	literal	0 - 3	Literal	S^#literal	r	r	r	r	r	-		L
4	PC	4	Immediate	I [PC]	r	r	r	r	r	f	r	f
5	Rn	5	Register	Rn	r	r	r	r	r	n	no	f
6	(Rn)	6	Register deferred	(Rn)	r	r	r	r	r	-	r	r
7	-(Rn)	7	Autodecrement	-(Rn)	r	r	r	r	r	n	r	no
8	(Rn)+	8	Autoincrement	(Rn)+	r	r	r	r	r	-	r	no
9	@(Rn)+	9	Autoincrement deferred	@(Rn)+	r	r	r	r	r	n	r	no
			displacement									
A	nn(Rn)	A	Byte displacement	nn(Rn)	r	r	r	r	r	n	r	r
B	@nn(Rn)	B	Byte displacement deferred	@nn(Rn)	r	r	r	r	r	n	r	r
C	nn(Rn)	C	Word displacement	nn(Rn)	r	r	r	r	r	n	r	r
D	@nn(Rn)	D	Word displacement deferred	@nn(Rn)	r	r	r	r	r	n	r	r
E	nn(Rn)	E	Longword displacement	nn(Rn)	r	r	r	r	r	n	r	r
F	@nn(Rn)	F	Longword displacement deferred	@nn(Rn)	r	r	r	r	r	n	r	r

PROGRAM COUNTER ADDRESSING

Hex		Dec	Name	Assembler	r m w a v	PC	SP	Indexable?
8	PC	8	immediate	I^#constant	y u u y y			y
9	PC	9	absolute	@#address	y y y y y	-		y
A	PC	10	byte relative	B^address	y y y y y			y
B	PC	11	byte relative deferred	@B^address	y y y y y	-	-	y
C	PC	12	word relative	W^address	y y y y y			y
D	PC	13	word relative deferred	@W^address	y y y y y			y
E	PC	14	longword relative	L^address	y y y y y	-		y
F	PC	15	longword relative deferred	@L^address	y y y y y	-	-	y

d = displacement
i = any indexed addressing mode
- = logically impossible
r = reserved addressing mode fault
p = Program Counter addressing
u = unpredictable
uq Unpredictable for quad and double (and field if position + size greater than 32)
ux Unpredictable for index register same as base register
y = yes, always valid addressing mode
r read access
m modify access
w write access
a address access
v field access

SPECIAL REGISTER USAGE

Register	Hardware Use	Conventional Software Use
R0	Builds of POINT,CRD; length counter in character & decimal instructions	Results of functions, status of services (not saved or restored on procedure call)
R1	Result of EDMD; address counter in character & decimal instructions	Result of functions (not saved or restored on procedure call)
R2, R3	Length counter in character & decimal instructions	Any
R3, R5	Bit or counter in character & decimal instructions	Any
R6-R11	None	Any
R12 (AP)	Argument pointer stack & loaded by CALL, restored by RET	Argument pointer (base address of argument list)
R13 (FP)	Frame pointer stack & loaded by CALL, used & restored by RET	Frame pointer (activation record)
R14 (SP)	Stack pointer	Stack pointer
R15 (PC)	Program counter	Program counter

bits 7:4 \ bits 3:0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	HALT	NOP	REI	BPT	RET	RSB	LDPCTX	SVPCTX	CVTPS	CVTSP	INDEX	CRC	PROBER	PROBEW	INSQUE	REMQUE
1	BSBB	BRB	BNEQ	BEQL	BGTR	BLEQ	JSB	JMP	BGEQ	BLSS	BGTRU	BLEQU	BVC	BVS	BCC	BCS
2	ADDP4	ADDP6	SUBP4	SUBP6	CVTPT	MULP	CVTTP	DIVP	MOVC3	MOVC5	CMPC3	CMPC5	MOVTC	MOVTUC	BUGW	BUGL
3	CASEB	CASEW	CASEL	Reserved	MOVAB	MOVAW	MOVAL	MOVAQ	ADDB2	ADDB3	ADDW2	ADDW3	ADDL2	ADDL3	SUBB2	SUBB3
4	MOVZBL	MOVZBW	MOVZWL	MOVZWW	MCOMB	MCOMW	MCOML	MNEGB	CALLG	CALLS	XFC	Reserved	Reserved	Reserved	Reserved	Reserved
5	INCB	INCW	INCL	DECB	DECW	DECL	TSTB	TSTW	BISPSW	BICPSW	POPR	PUSHR	CHMK	CHME	CHMS	CHMU
6	ADDB	ADDW	ADDL	SUBB	SUBW	SUBL	ASHP	CVTLP	ACBB	ACBW	ACBL	AOBLSS	AOBLEQ	SOBGEQ	SOBGTR	CVTLB
7	MULB	MULW	MULL	DIVB	DIVW	DIVL	CMPV	CMPZV	FFS	FFC	CMPW	MOVP	CMPP3	CMPP4	EDITPC	MATCHC
8	BISB2	BISB3	BICB2	BICB3	BISW2	BISW3	BICW2	BICW3	BISL2	BISL3	BICL2	BICL3	XORB2	XORB3	XORW2	XORW3
9	XORL2	XORL3	MOVN	CMPC	MOVZ	CVTPL	CMPV	CMPZV	EMODP	Reserved	CVTWL	CVTWB	MOVPSL	PUSHAB	PUSHAW	PUSHAL
A	ADDF2	ADDF3	SUBF2	SUBF3	MULF2	MULF3	DIVF2	DIVF3	CVTFB	CVTFW	CVTFL	CVTRFL	CVTBF	CVTWF	CVTLF	ACBF
B	MOVF	CMPF	MNEGF	TSTF	EMODF	POLYF	CVTFD	Reserved	ADAWI	Reserved	INSQHI	REMQHI	INSQTI	REMQTI	INSQUED	REMQUED
C	ADDD2	ADDD3	SUBD2	SUBD3	MULD2	MULD3	DIVD2	DIVD3	CVTDB	CVTDW	CVTDL	CVTRDL	CVTBD	CVTWD	CVTLD	ACBD
D	MOVD	CMPD	MNEGD	TSTD	EMODD	POLYD	CVTDF	Reserved	ASHQ	EMUL	EDIV	CLRQ	MOVQ	MOVAQ	PUSHAQ	ADWC
E	ADDG2	ADDG3	SUBG2	SUBG3	MULG2	MULG3	DIVG2	DIVG3	CVTGB	CVTGW	CVTGL	CVTRGL	CVTBG	CVTWG	CVTLG	ACBG
F	MOVG	CMPG	MNEGG	TSTG	EMODG	POLYG	CVTGF	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

VAX-11 INSTRUCTION OPERAND SPECIFIER NOTATION

OPERAND SPECIFIERS ARE SPECIFIED IN THE FOLLOWING MANNER:

<NAME>.<ACCESS TYPE><DATA TYPE>

1. NAME --- SUGGESTIVE NAME FOR OPERAND IN THE CONTEXT OF THE INSTRUCTION

2. ACCESS TYPE --- LETTER DENOTING OPERAND SPECIFIER ACCESS TYPE:

   A - CALCULATE THE EFFECTIVE ADDRESS OF THE SPECIFIED OPERAND. ADDRESS IS RETURNED IN A LONGWORD WHICH IS THE ACTUAL INSTRUCTION OPERAND. CONTENT OF ADDRESS CALCULATION IS GIVEN BY <DATA TYPE>.

   B - NO OPERAND REFERENCE. OPERAND SPECIFIER IS A BRANCH DISPLACEMENT. SIZE OF BRANCH DISPLACEMENT IS GIVEN BY <DATA TYPE>.

   M - OPERAND IS READ, POTENTIALLY MODIFIED AND WRITTEN. THIS IS NOT AN INDIVISIBLE MEMORY OPERATION.

   R - OPERAND IS READ ONLY.

   W - OPERAND IS WRITE ONLY.

3. DATA TYPE --- IS A LETTER DENOTING THE DATA TYPE OF THE OPERAND:

   B - BYTE

   D - DOUBLE FLOATING

   F - FLOATING

   L - LONG WORD

   Q - QUAD WORD

   W - WORD

INSTRUCTIONS	N	Z	V	C
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

INSTRUCTION	N	Z	V	C
[illegible]	?	?	?	?
[illegible]	?	?	?	?
ADDG - ADD G_FLOATING ADDG2,3 src.rg, [dst.wg,] dst.wg	?	?	?	?
CMPG - COMPARE G_FLOATING CMPG src1.rg, src2.rg	?	?	0	?
CVTGB - CVT G_FLOATING TO BYTE CVTGB src.rg, dst.wb	?	?	?	?
CVTGF - CVT G_FLOATING TO F_FLOATING CVTGF src.rg, dst.wf	?	?	?	?
CVTGH - CVT G_FLOATING TO H_FLOATING CVTGH src.rg, dst.wh	?	?	?	?
CVTGL - CVT G_FLOATING TO LONG CVTGL src.rg, dst.wl	?	?	?	?
CVTGW - CVT G_FLOATING TO WORD CVTGW src.rg, dst.ww	?	?	?	?
CVTHG - CVT H_FLOATING TO G_FLOATING CVTHG src.rh, dst.wg	?	?	?	?
CVTLG - CVT LONG TO G_FLOATING CVTLG src.rl, dst.wg	?	?	?	?
CVTRGL - CVT ROUNDED G_FLOATING TO LONG CVTRGL src.rg, dst.wl	?	?	?	?
DIVG - DIVIDE G_FLOATING DIVG2,3 src.rg, [dst.mg,] dst.wg	?	?	?	?
EMODG - EXTENDED MULTIPLY AND INTEGERIZE G_FLOATING EMODG mulr.rg, mulx.rl, add.rg, intd.wl, fracd.wg	?	?	?	?
MNEGG - MOVE NEGATED G_FLOATING MNEGG src.rg, dst.wg	?	?	?	?
MOVG - MOVE G_FLOATING MOVG src.rg, dst.wg	?	?	0	-
MULG - MULTIPLY G_FLOATING MULG2,3 src.rg, [dst.mg,] dst.wg	?	?	?	?
POLYG - EVALUATE POLYNOMIAL G_FLOATING POLYG arg.rg, degree.rw, table.ab	?	?	?	?
TSTG - TEST G_FLOATING TSTG src.rg	?	?	0	0

INSTRUCTION	N	CONDITION CODES Z	V	C
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]				

BRANCH CONDITIONS

OPCODE	CONDITIONS
BGT	N or Z = 0
BLE	N or Z = 1
BNE	Z = 0
BNEQU	Z = 0
BEQL	Z = 1
BEQLU	Z = 1
BGE	N = 0
BLSS	N = 1
BGTU	C or Z = 0
BLEU	C or Z = 1
BVC	V = 0
BVS	V = 1
BGEQU	C = 0
BCC	C = 0
BLSSU	C = 1
BCS	

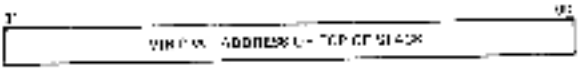
VAX-11/780 PROCESSOR REGISTER ADDRESSES

HEX	DEC			
00	0	KSP	Kernel stack pointer	
01	1	ESP	Executive stack pointer	
02	2	SSP	Supervisor stack pointer	
03	3	USP	User stack pointer	
04	4	ISP	Interrupt stack pointer	
05	5	reserved		
06	6	reserved		
07	7	reserved		
08	8	P0BR	P0 base register	
09	9	P0LR	P0 length register	
0A	10	P1BR	P1 base register	
0B	11	P1LR	P1 length register	
0C	12	SBR	System base register	
0D	13	SLR	System length register	
0E	14	reserved		
0F	15	reserved		
10	16	PCBB	Process control block base	
11	17	SCBB	System control block base	
12	18	IPL	Interrupt priority level	
13	19	ASTLVL	AST level register	
14	20	SIRR	Software interrupt request register	WO
15	21	SISR	Software interrupt summary register	
16	22	reserved		
17	23	reserved		
18	24	ICCS	Interval clock control/status	
19	25	NICR	Next interval count register	WO
1A	26	ICR	Interval count register	RO
1B	27	TODR	Time of day register	
1C	28	reserved		
1D	29	reserved		
1E	30	reserved		
1F	31	reserved		
20	32	RXCS	Console receive control/status	
21	33	RXDB	Console receive data buffer	RO
22	34	TXCS	Console transmit control/status	
23	35	TXDB	Console transmit data buffer	WO
24	36	reserved		
25	37	reserved		
26	38	reserved		
27	39	reserved		
28	40	ACCS	Accelerator control/status	
29	41	ACCR	Accelerator reserved	
2A	42	reserved		
2B	43	reserved		
2C	44	WCSA	Writable control store address	
2D	45	WCSD	Writable control store data	
2E	46	reserved		
2F	47	reserved		
30	48	SBIFS	SBI fault/status	
31	49	SBIS	SBI silo	RO
32	50	SBISC	SBI silo comparator	
33	51	SBIMT	SBI maintenance	
34	52	SBIER	SBI error register	
35	53	SBITA	SBI timeout address	RO
36	54	SBIQC	SBI quadword clear	WO
37	55	reserved		
38	56	MAR	Memory array wait enable	
39	57	MDCR	Translation buffer invalidate all	WO
3A	58	TBIA	Translation buffer invalidate single	WO
3B	59	reserved		
3C	60	MBRK	Microprogram breakpoint	
3D	61	PME	Performance monitor enable	
3E	62	SID	System identification	RO
3F	63	reserved		

VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

KPSP /
DECIMAL VALUE OF

0	00	KSP	30	KERNEL STACK POINTER
1	01	ESP	29	EXECUTIVE STACK POINTER
2	10	SSP	28	SUPERVISOR STACK POINTER
3	11	USP	27	USER STACK POINTER
4	04	ISP	26	INTERRUPT STACK POINTER



8 08 P0BR 24 P0 BASE REGISTER

RESERVED OPERAND FAULT IF VIRTUAL

10 09 P1BR 25 P1 BASE REGISTER

RESERVED OPERAND FAULT IF VIRTUAL

9 09 P0LR 26 P0 LENGTH REGISTER

RESERVED TO 0 IN LONGWORD

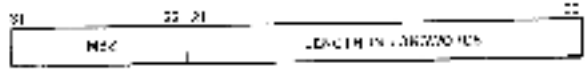
A 0A P1LR 27 P1 LENGTH REGISTER

RESERVED TO 0 IN LONGWORD

B 0B SLR 28 SYSTEM LENGTH REGISTER

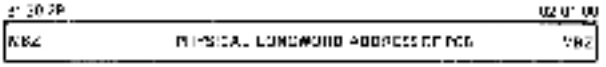
LENGTH OF SYSTEM PAGE TABLE

RESERVED OPERAND FAULT IF VAL

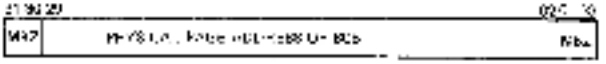


REG #
DEC HEX NAME D*

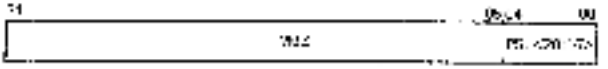
16 10 7810 3A PROCESS CONTROL BLOCK BASE
RESERVED OPERAND FAULT IF PMR = 0



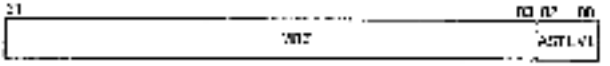
17 11 8811 3B SYSTEM CONTROL BLOCK BASE
RESERVED OPERAND FAULT IF MBZ ≠ 0



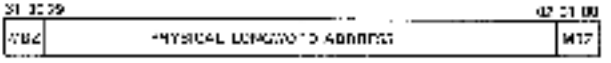
18 12 7812 3C INTERRUPT PRIORITY LEVEL REGISTER

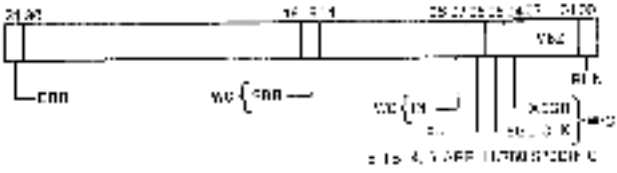


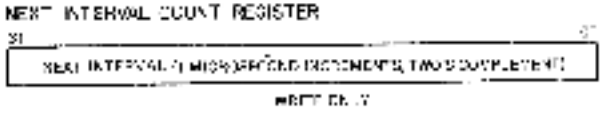
19 13 7813 3D ASTLVL - AST LEVEL REGISTER
RESERVED OPERAND FAULT IF NOT VALID AST LEVEL

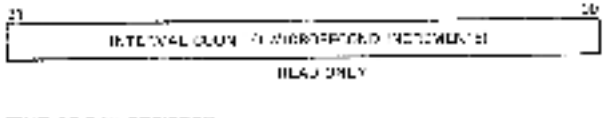


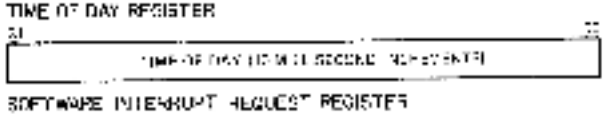
20 14 8814 SYSTEM BASE REGISTER
RESERVED OPERAND FAULT IF MBZ ≠ 0

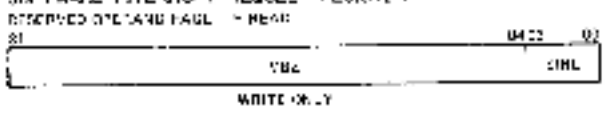


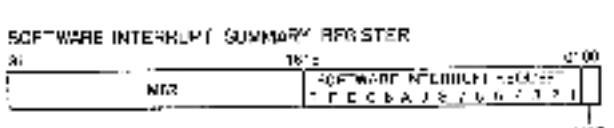


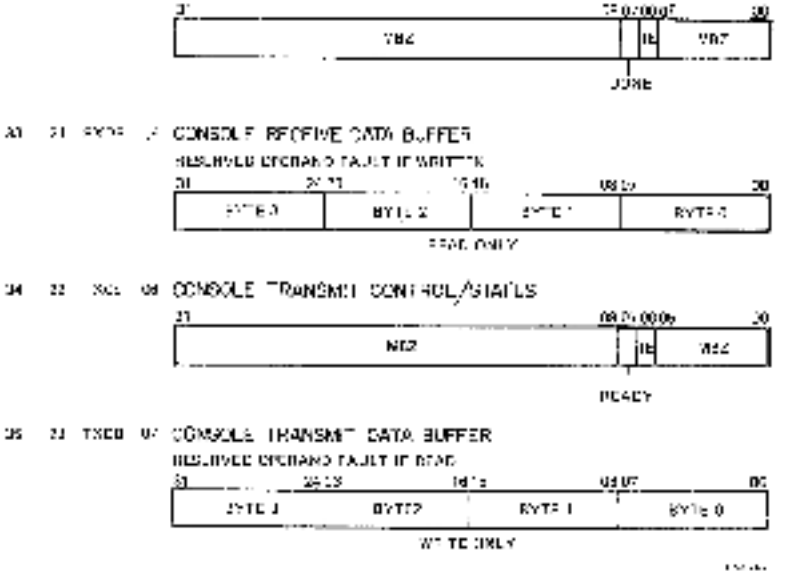




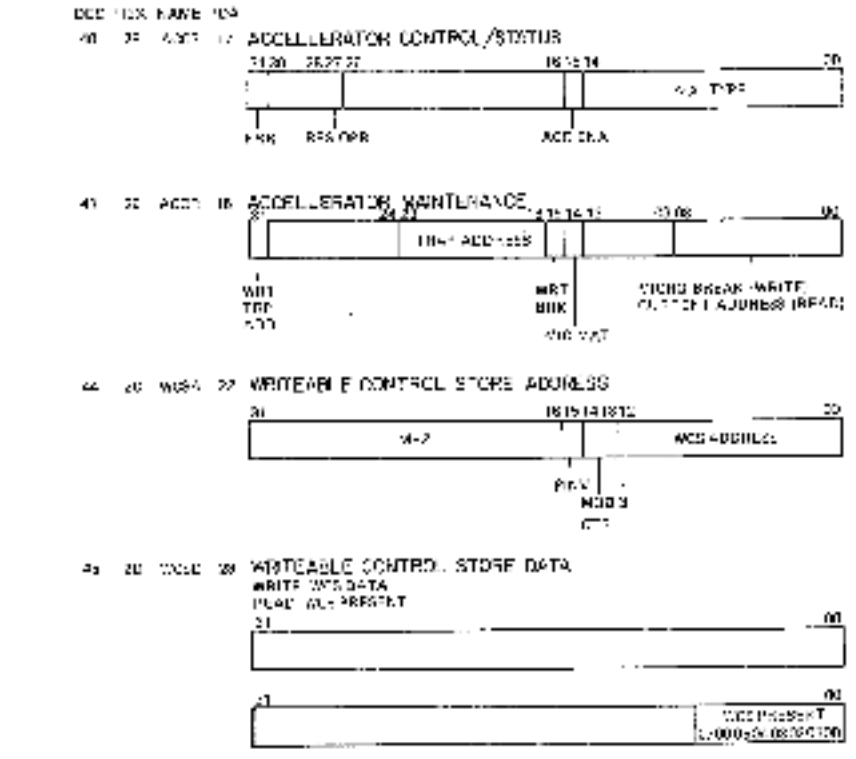




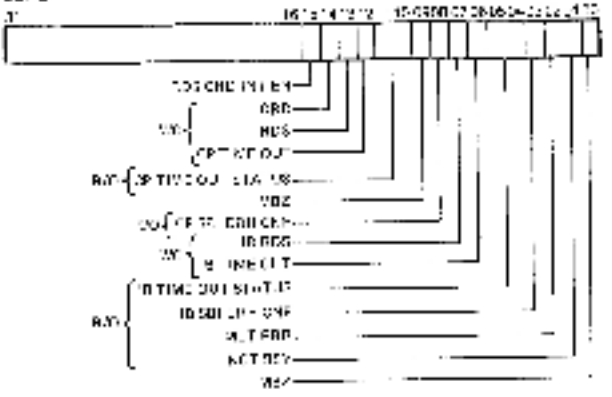


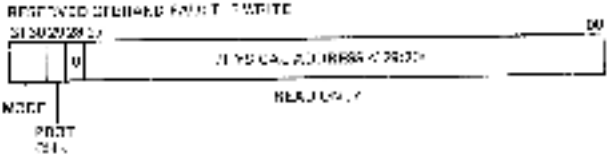

31
07
06
00
IE
DONE
31
00
BYTE 3
BYTE 2
BYTE 1
BYTE 0
READ ONLY
31
07
06
00
IE
READY
31
15
14
08
07
00
BYTE 1
BYTE 0
WRITE ONLY

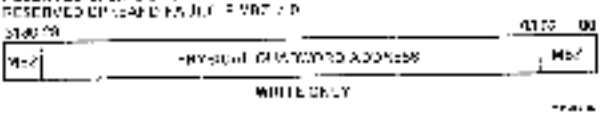
VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

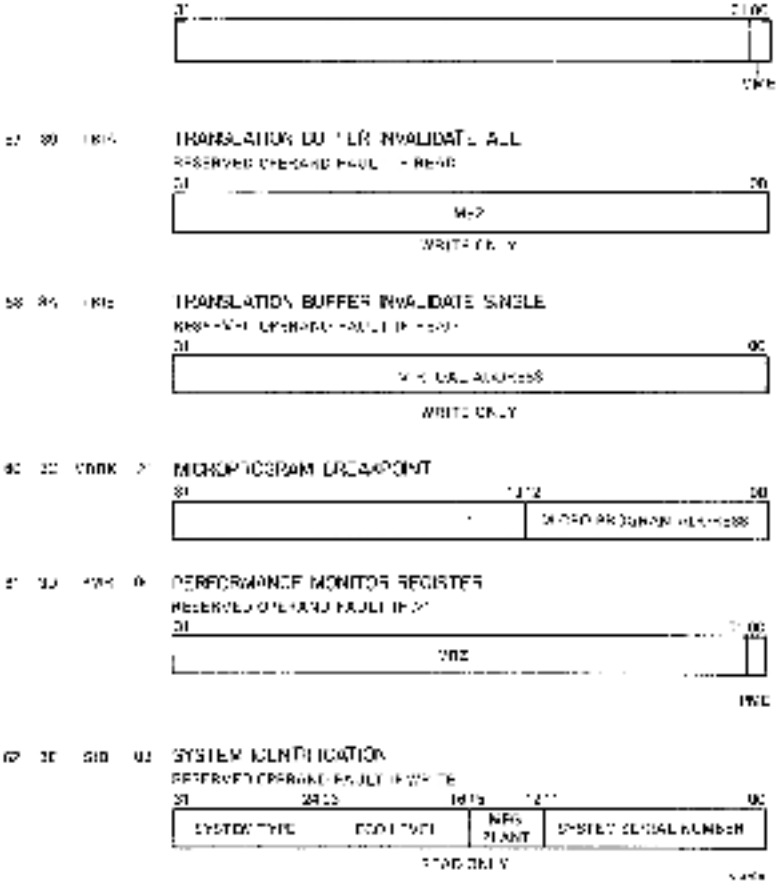


VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS








MEMORY MANAGEMENT ENABLE
TRANSLATION BUFFER INVALIDATE ALL
RESERVED OPERAND FAULT
WRITE ONLY
TRANSLATION BUFFER INVALIDATE SINGLE
RESERVED OPERAND FAULT
WRITE ONLY
MICROPROGRAM BREAKPOINT
PERFORMANCE MONITOR REGISTER
RESERVED OPERAND FAULT
SYSTEM IDENTIFICATION
RESERVED OPERAND FAULT
SYSTEM TYPE
SYSTEM SERIAL NUMBER
READ ONLY

```
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

SYSTEM CONTROL BLOCK BASE

+---------------------------------------------------------------+---------------------------------+
|                                                               |                                 |
+---------------------------------------------------------------+---------------------------------+

VECTORED INTERRUPTS

  0   RESERVED
  1   RESERVED
  2   NMI
  3   SERVICE REQUEST BASE DATE (BIT 30 RESET)
  4   RESERVED
  5   RESERVED
  6   RESERVED
  7   RESERVED
  8   RESERVED
  9   RESERVED
 10   RESERVED
 11   RESERVED
 12   RESERVED
 13   TRACE (T=1)
 14   DEBUGPOINT
 15   INCOMPATIBILITY
 16   ARITHMETIC
 17   RESERVED
 18   CHK
 19   CHKA
 20   CHKR
 21   CHKI
 22   RESERVED
 23   CHK(I)(A)
 24   SSC ALERT
 25   SSC FAULT
 26   CPI REQUEST (MPSC RESYNCHRONIZE WRITE TIMECLK)
 27   RESERVED
 28   SOFTWARE LEVEL 1
 29   SOFTWARE LEVEL 2
 30   SOFTWARE LEVEL 3
 31   INTERVAL TIMER
 32   RESERVED
 33   CHK
 34   RESERVED
```

```
CHECKSUMS <BLOCK>
CHECKSUMS VERIFY
CHECKSUMS <BLOCK>
CHECKSUMS VERIFY
```

```
*100 132  201 R/W 1
*101-115  -R  R-O 1
*116-131  -RW R-O 1
*201-215  -R  R-O 1

** = volume has [1;"3f specific
** = vo  d  v  v  v  1  -RW.
  RE = RO in db db d vers 1 CHECKSUM volume changed.
```


SCSI II Mode Sense Codes

CODE	DESCRIPTION
1	M[illegible] OVERRIDE SENSE
2	CHECKSUM DISTRO BY ZERO TRAP
3	PIGHTING DIVISION TRAP
4	PIGHTING DEVICE BY SIZE TRAP
5	PIGHT M UNDISPLAY TRAP
6	M[illegible] [illegible] IN [illegible]
7	M[illegible] [illegible] BY ZERO TRAP

Compatibility mode codes

CODE	DESCRIPTION
0	[illegible]
1	[illegible]
2	ICC
3	CHK
4	SCAC
5	ILLEGAL [illegible]
6	SEE ADDRESS

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

PROCESS CONTROL BLOCK ADDR (PCB)

PHYSICAL LOAD AREA ADDRESS OF PCB	ABS

PROCESS CONTROL BLOCK (PCB)

	RSP	TCB +0
	RSP	+4
	RSP	+8
	USP	+C
	R2	+10
	R3	+14
	R4	+18
	R5	+1C
	R6	+20
	R7	+24
	R8	+28
	R9	+2C
	R10	+30
	R11	+34
	R12	+38
	R13	+3C
	R14	+40
	PSW	+44
	PCB	+48
	TIOB	+4C
	PTR	+50

PROTECTION CODES

CODE	MEANING			
	K	E	S	U
0 0 0 0	*	*	*	*
0 0 0 1	UNPREDICTABLE			
0 0 1 0	R/W	*	*	*
0 0 1 1	RO	*	*	*
0 1 0 0	R/W	R/W	R/W	R/W
0 1 0 1	R/W	R/W	*	*
0 1 1 0	R/W	RO	*	*
0 1 1 1	RO	RO	*	*
1 0 0 0	R/W	R/W	R/W	*
1 0 0 1	R/W	R/W	RO	*
1 0 1 0	R/W	RO	RO	*
1 0 1 1	RO	RO	RO	*
1 1 0 0	R/W	R/W	R/W	RO
1 1 0 1	R/W	R/W	RO	RO
1 1 1 0	R/W	RO	RO	RO
1 1 1 1	RO	RO	RO	RO

K KERNEL
E EXECUTIVE
S SUPERVISOR
U USER

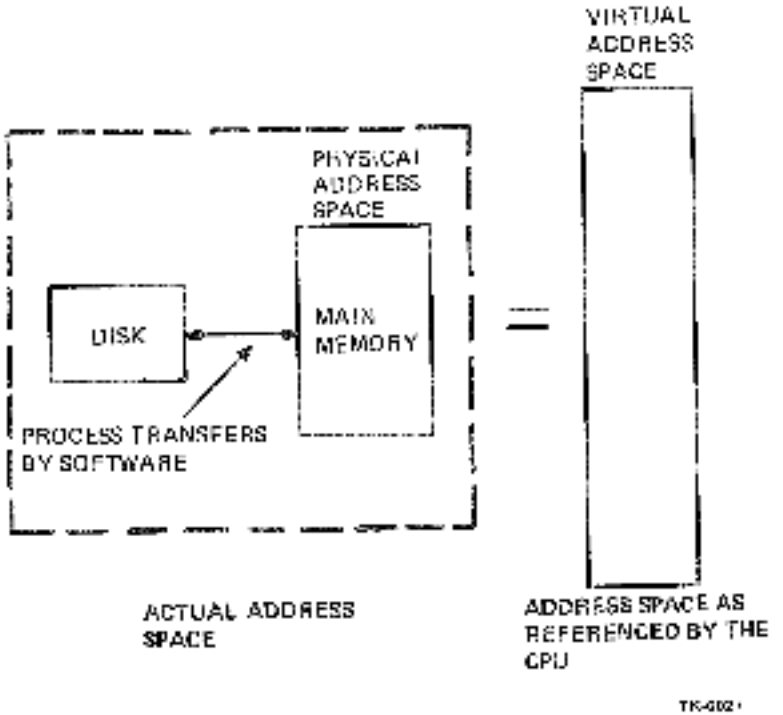
* NO ACCESS
RO READ ONLY
R/W READ WRITE

MODE

0	(00)	KERNEL
1	(01)	EXECUTIVE
2	(10)	SUPERVISOR
3	(11)	USER

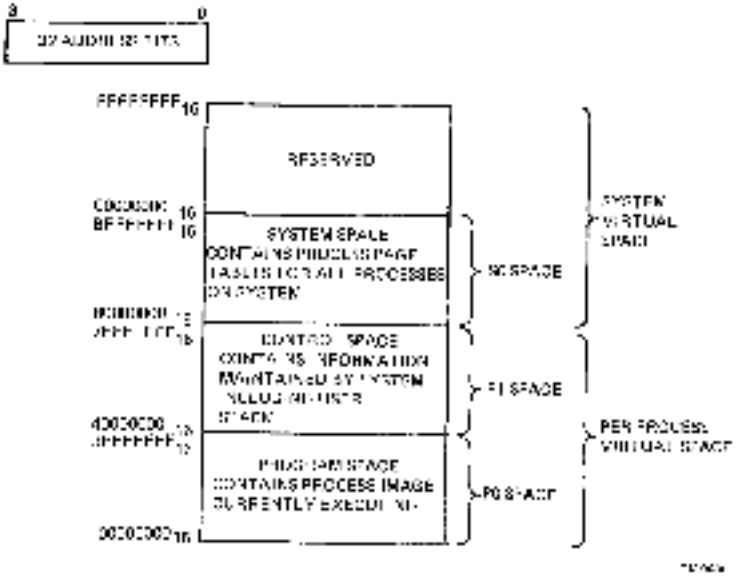
CONDITION	VECTOR
MACHINE CHECK	04
KERNEL STACK NOT VALID	08
RESERVED DEC OPCODES & PRIVILEGED INSTRUCTIONS	10
RESERVED CUSTOMER OPCODES	14
RESERVED OPERANDS	18
RESERVED ADDRESSING MODES	1C
ACCESS CONTROL VIOLATION	20
TRANSLATION NOT VALID	24
TRACE TRAP	28
BPT OPCODE	2C
COMPATABILITY MODE TRAP	30
ARITHMATIC TRAP	34
CHMK OPCODE	40
CHME OPCODE	44
CHMS OPCODE	48
CHMU OPCODE	4C

VIRTUAL AND PHYSICAL ADDRESS RELATIONSHIP

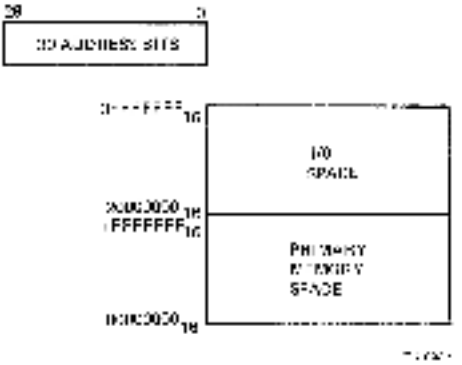


VIRTUAL AND PHYSICAL ADDRESS SPACE

VIRTUAL ADDRESS SPACE

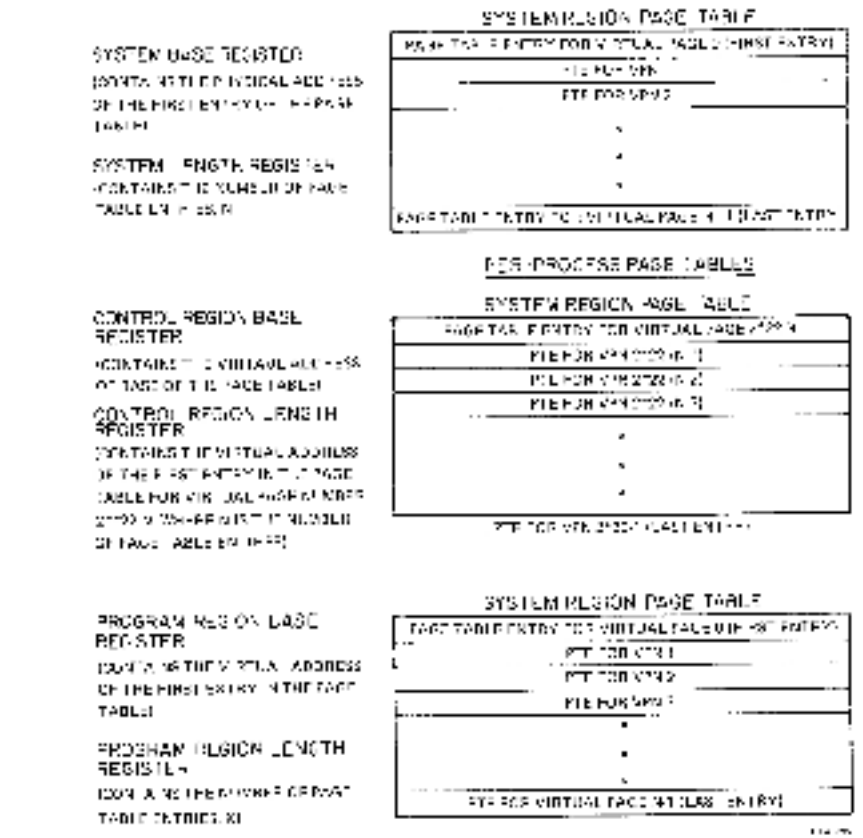


PHYSICAL ADDRESS SPACE

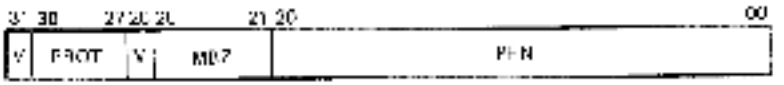


PAGE TABLE FORMATS AND PAGE TABLE ENTRY FORMAT

PAGE TABLE FORMATS

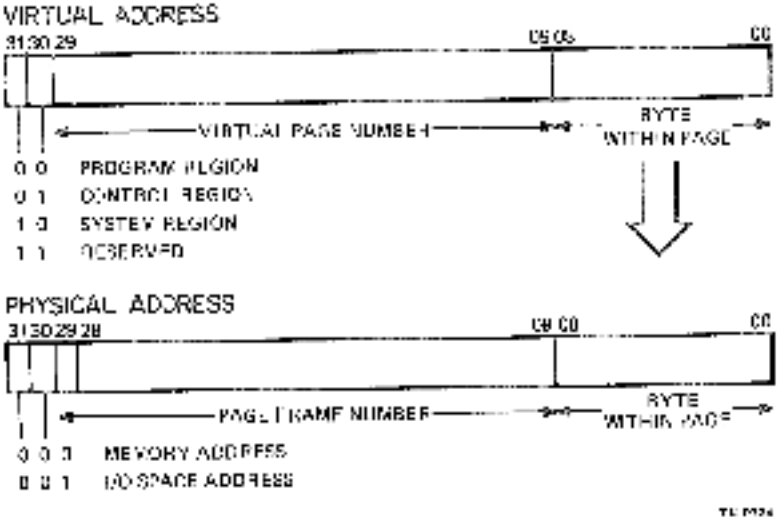


PAGE TABLE ENTRY FORMAT

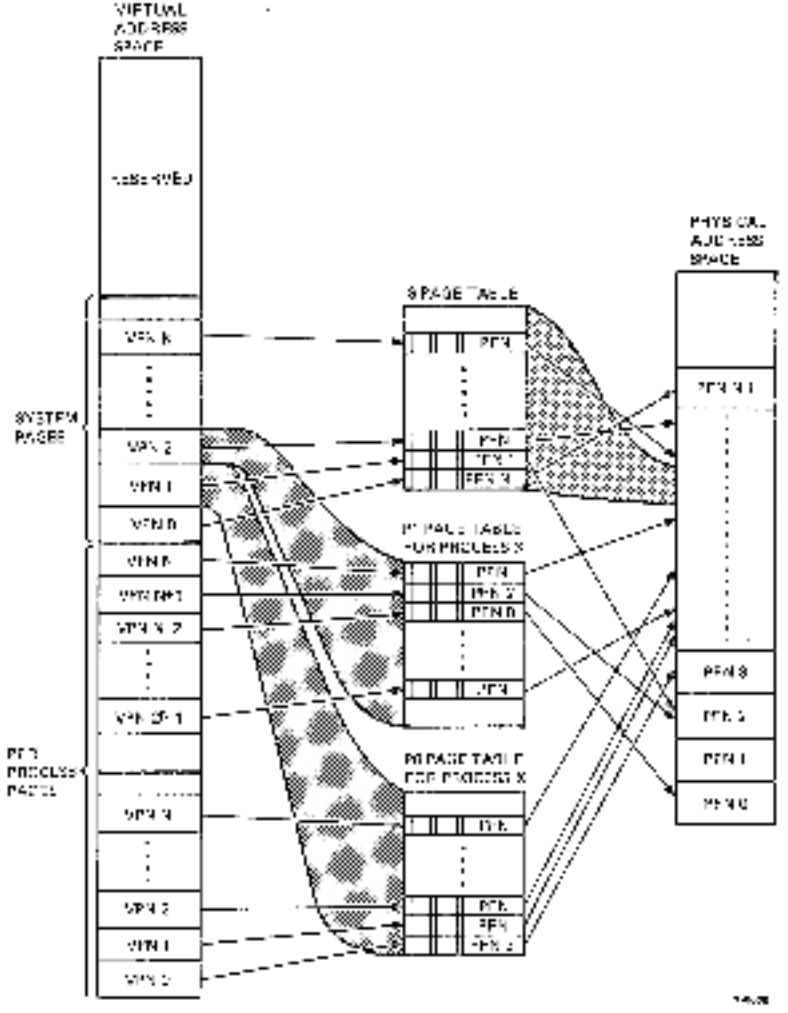


EXAMPLE OF PAGE FRAME ALLOCATION (RELOCATION)

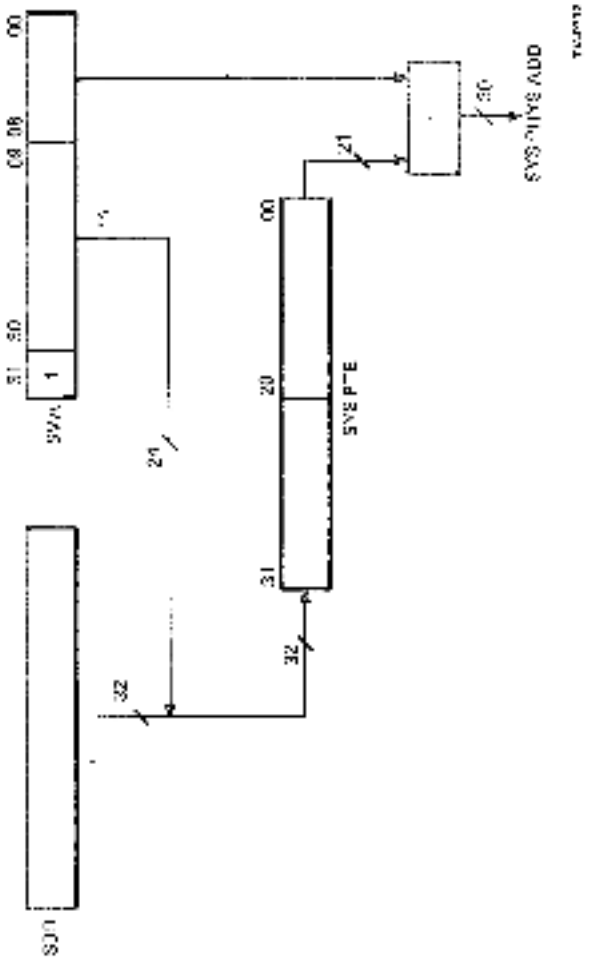
VIRTUAL AND PHYSICAL ADDRESS FORMATS



VIRTUAL PAGES MAPPED TO PHYSICAL SPACE

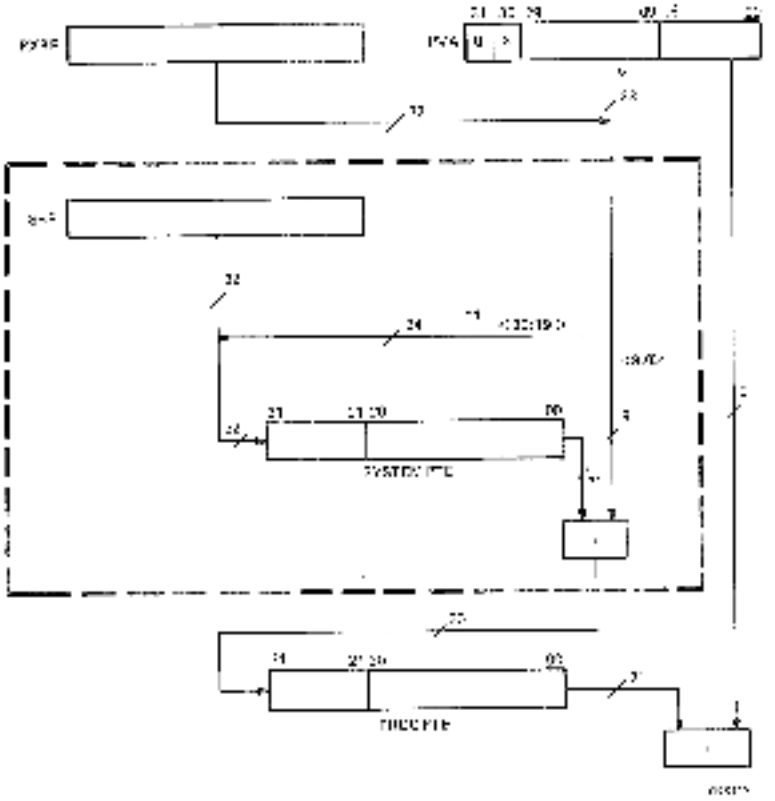


SYSTEM VIRTUAL TO PHYSICAL ADDRESS TRANSLATION SCHEME

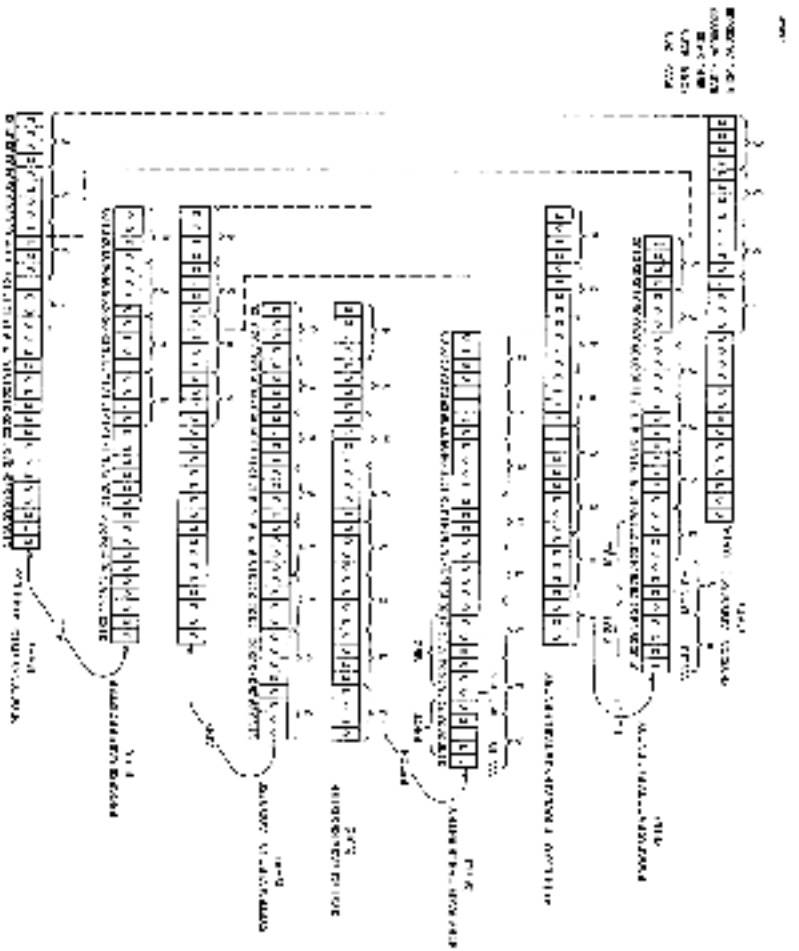


SYSTEM VIRTUAL TO PHYSICAL ADDRESS TRANSLATION, EXAMPLE

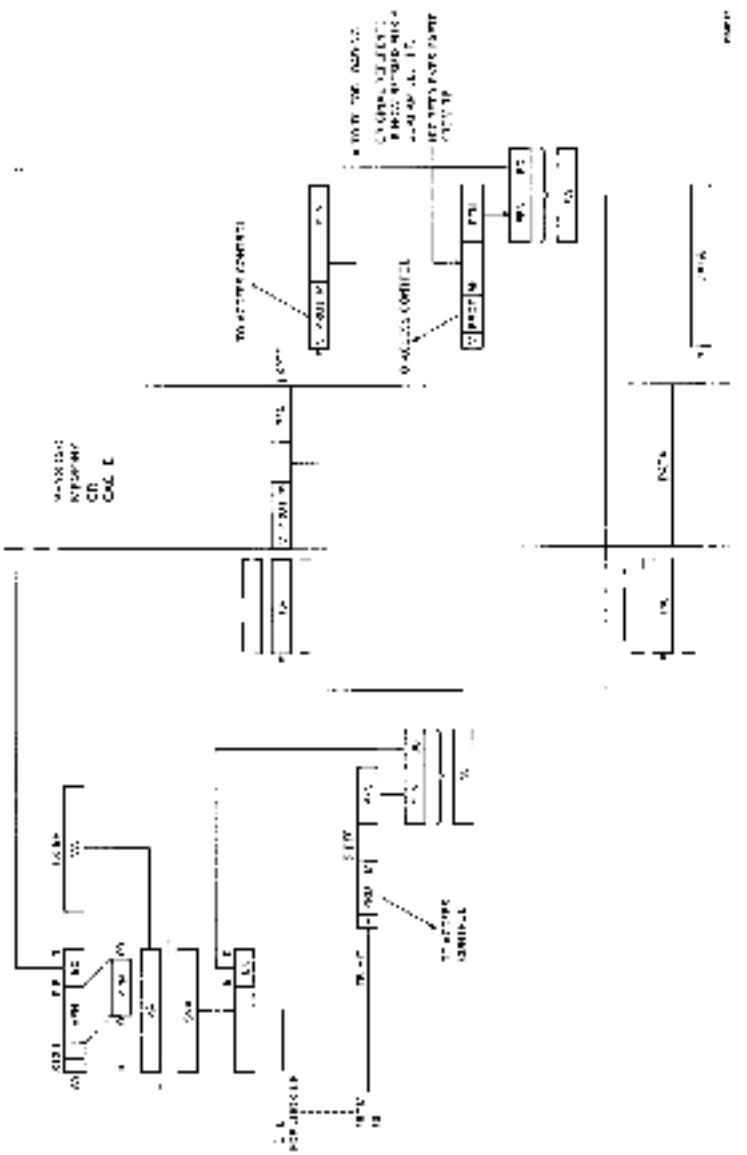
PROCESS VIRTUAL TO PHYSICAL ADDRESS TRANSLATION SCHEME



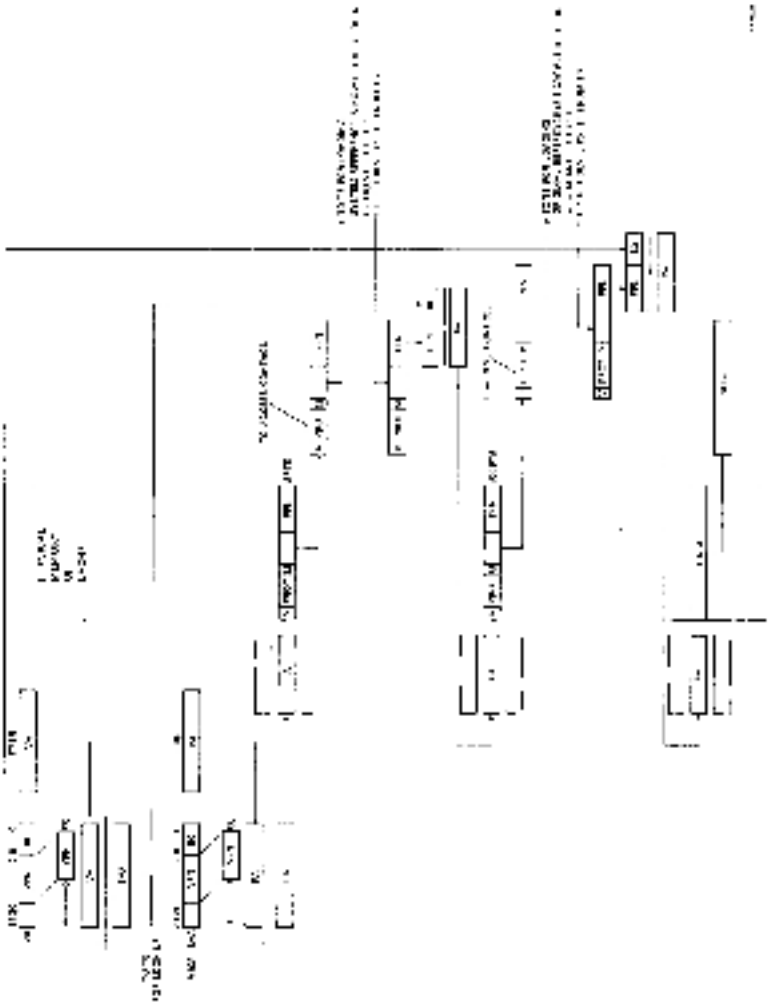
PROCESS VIRTUAL TO PHYSICAL ADDRESS TRANSLATION, EXAMPLE



ADDRESS CALCULATION FOR A TB HIT DURING A MISS MICROTRAP

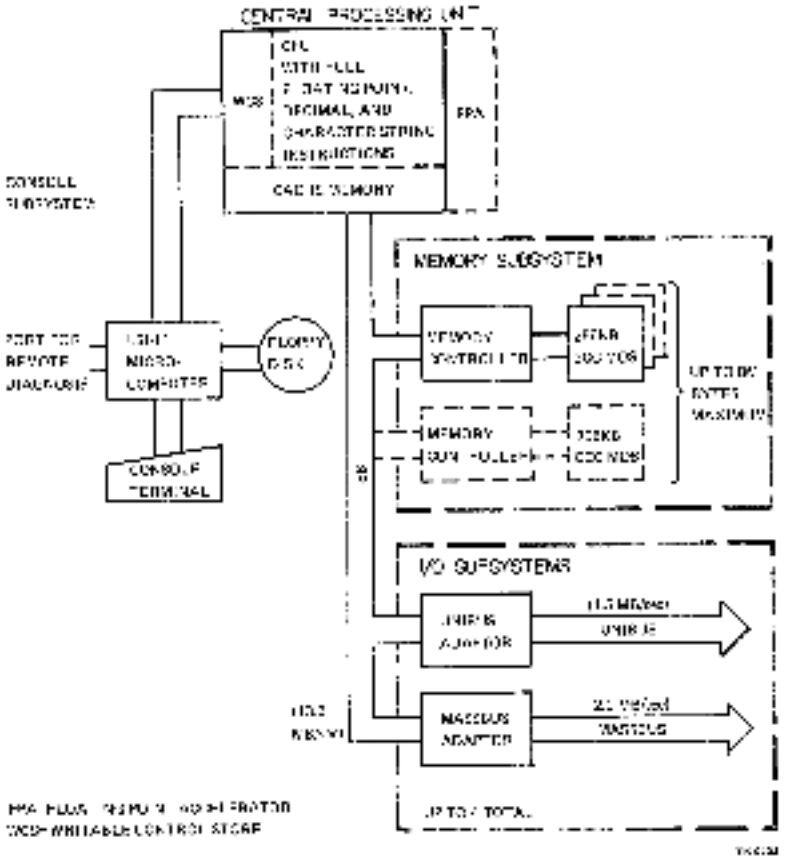


ADDRESS CALCULATION FOR A TB MISS DURING A MISS MICROTRAP



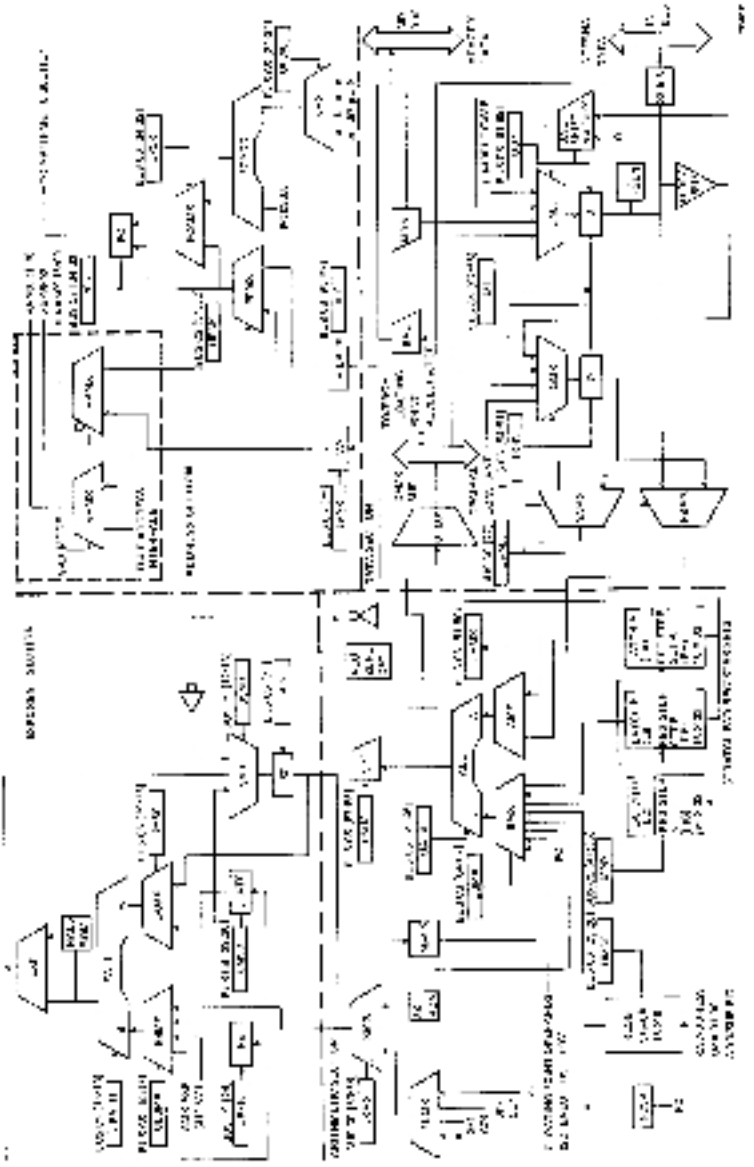
SECTION 3
HARDWARE BLOCK DIAGRAMS AND REGISTER BIT CONFIGURATIONS

VAX-11/780 GENERAL BLOCK DIAGRAM

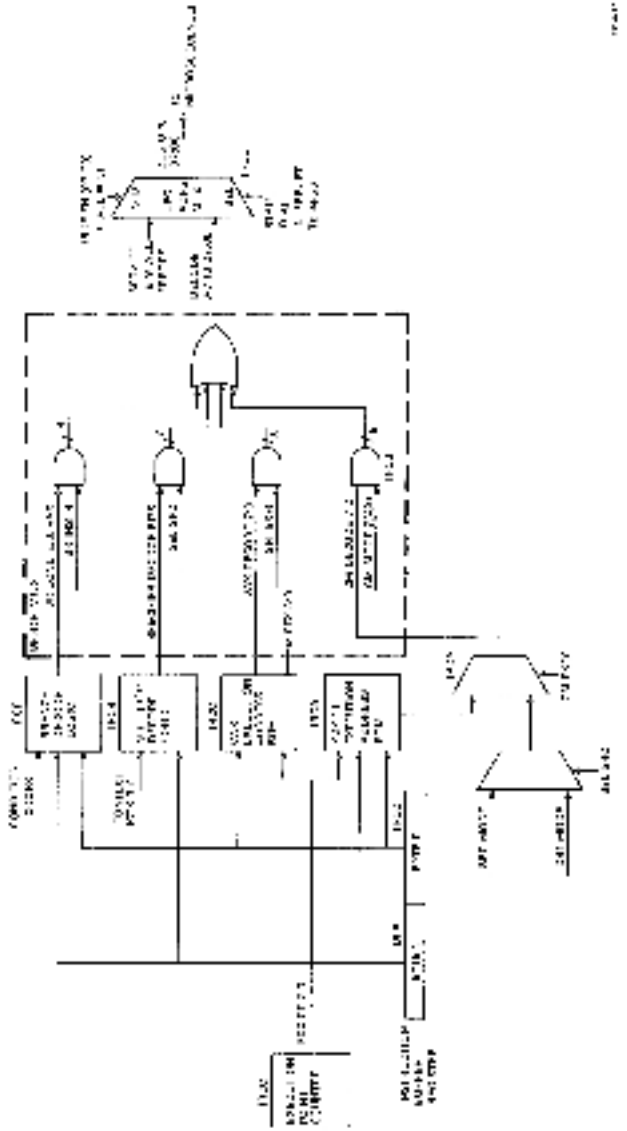


CPU BLOCK DIAGRAM

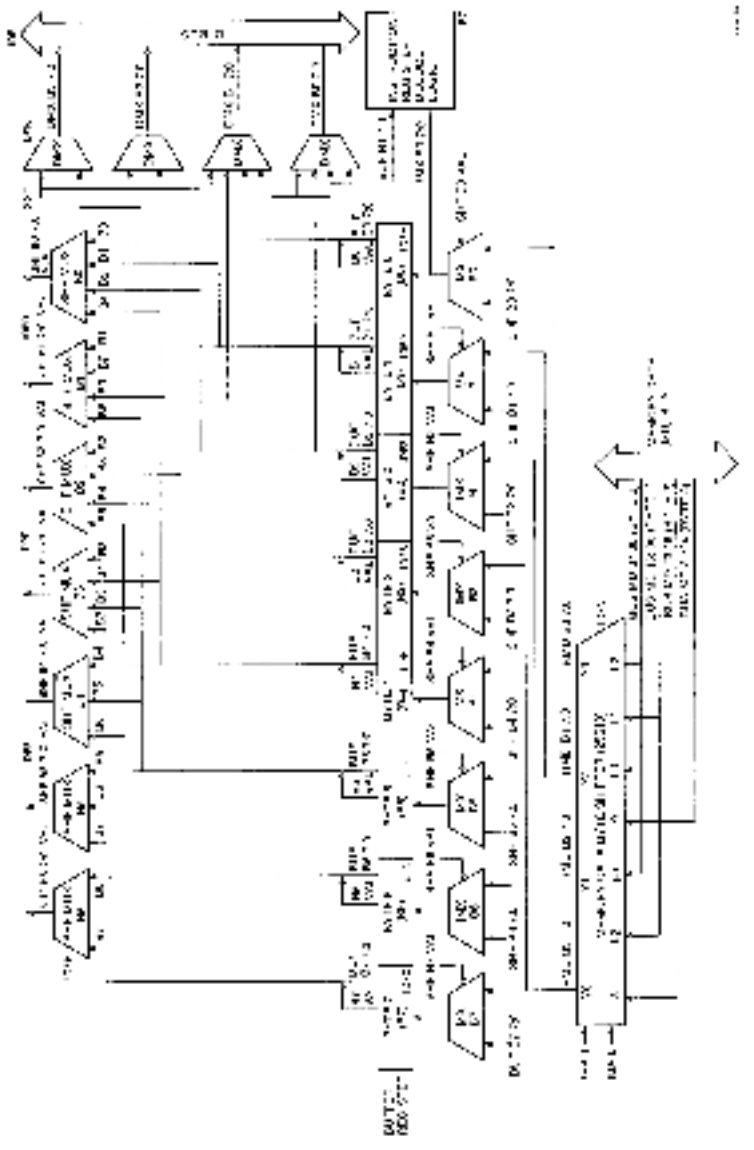
DATA PATH BLOCK DIAGRAM



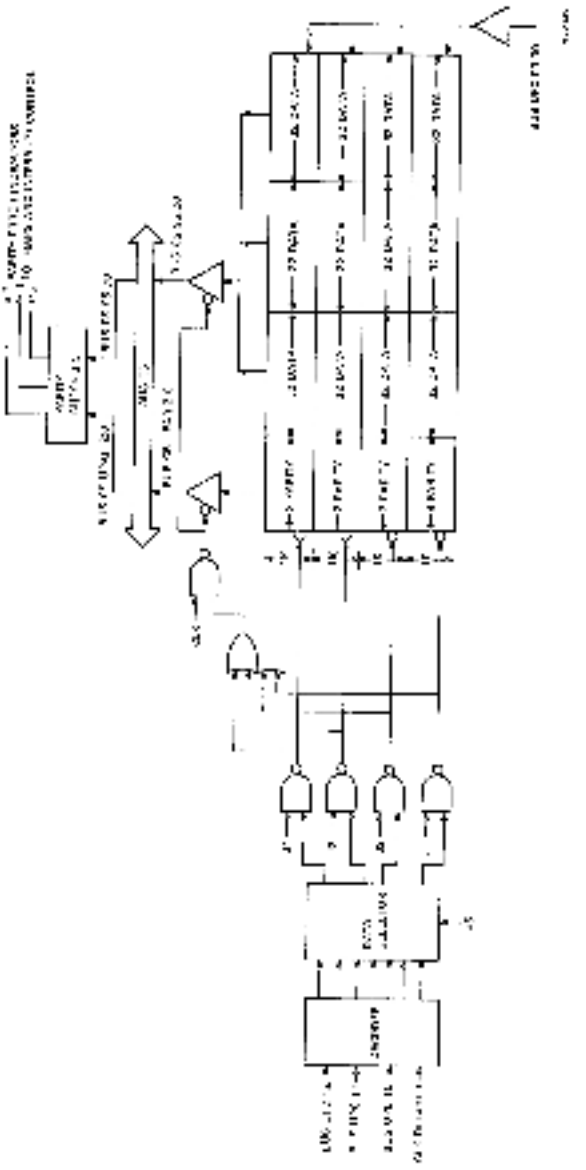
INSTRUCTION DECODE BLOCK DIAGRAM



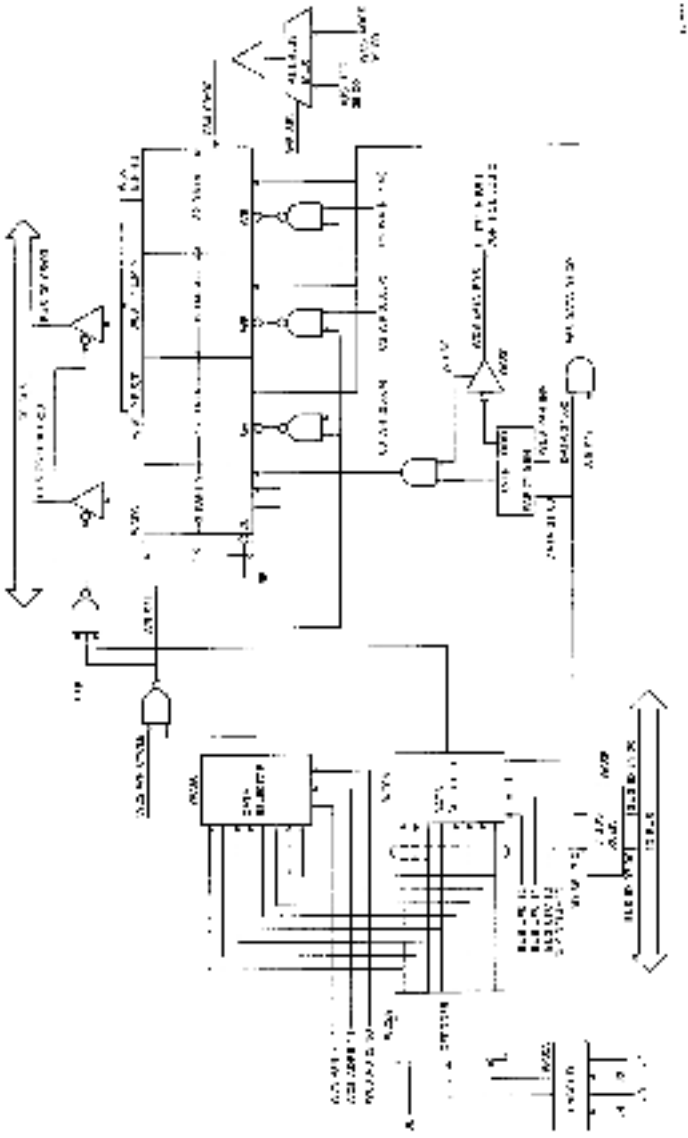
INSTRUCTION BUFFER BLOCK DIAGRAM



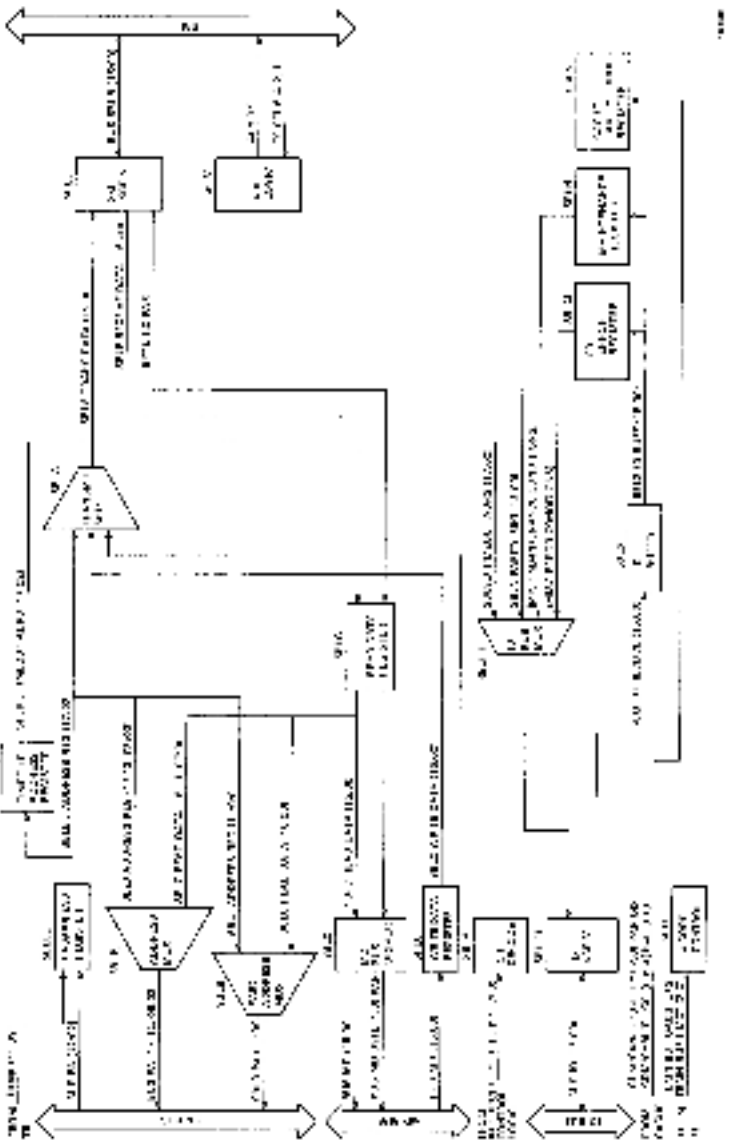
PROM CONTROL STORE (PCS) BLOCK DIAGRAM



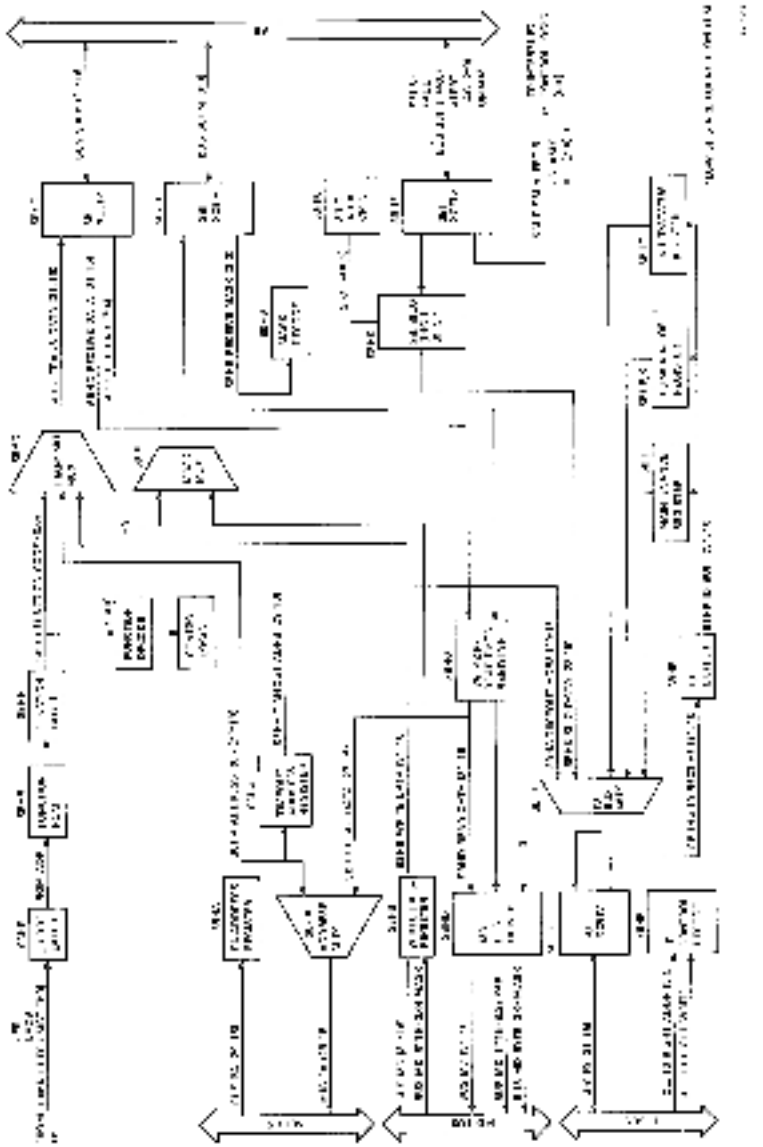
WRITABLE CONTROL STORE (WCS) BLOCK DIAGRAM



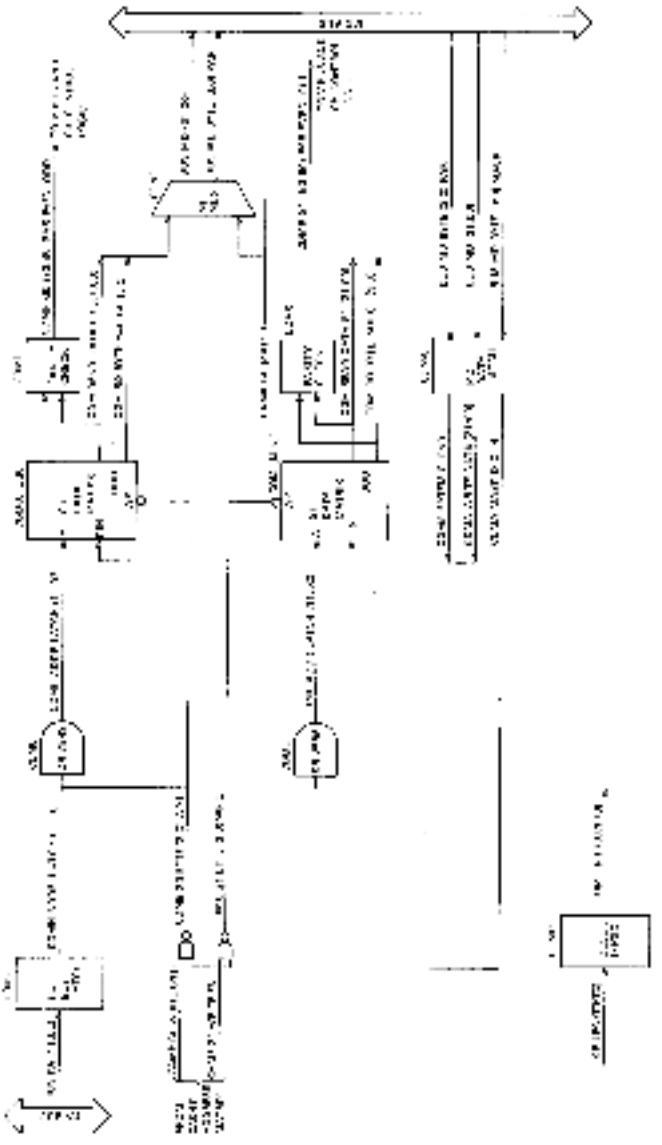
SBI CONTROL LOW BITS (SBL) BLOCK DIAGRAM



SBI CONTROL HIGH BITS (SBIC) BLOCK DIAGRAM

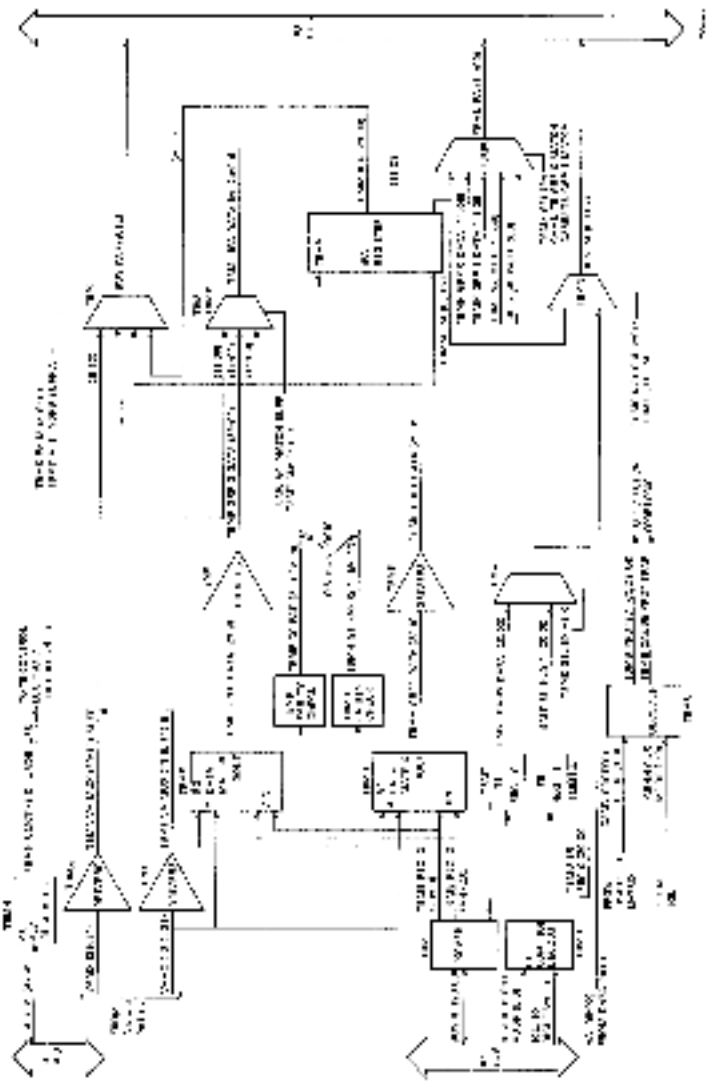


CACHE DATA MATRIX BLOCK DIAGRAM

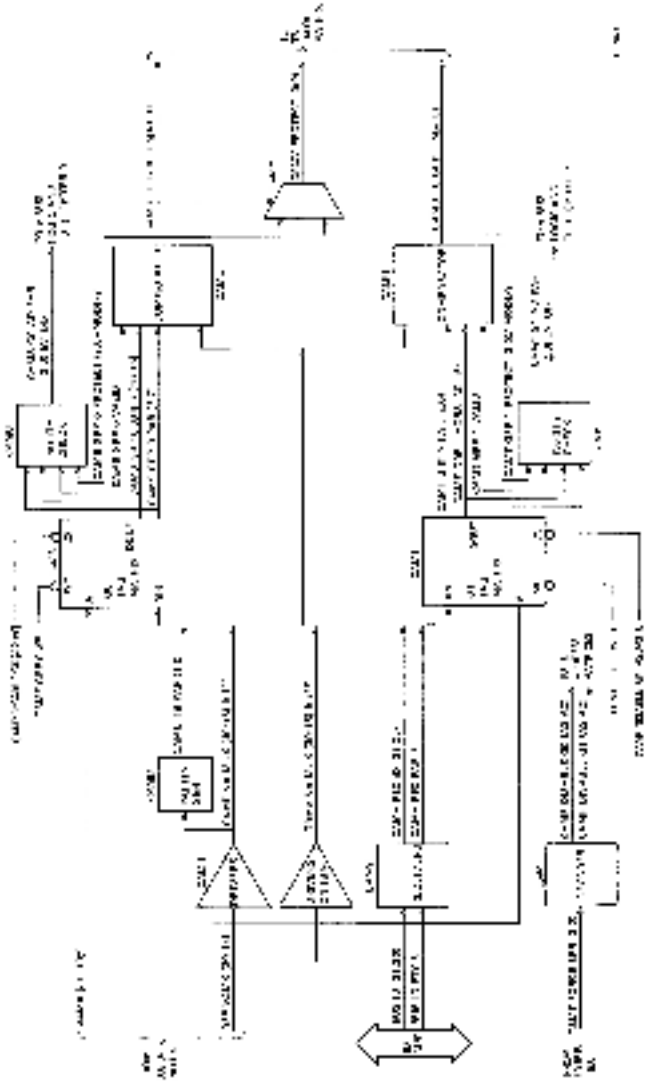


CACHE ADDRESS MATRIX BLOCK DIAGRAM

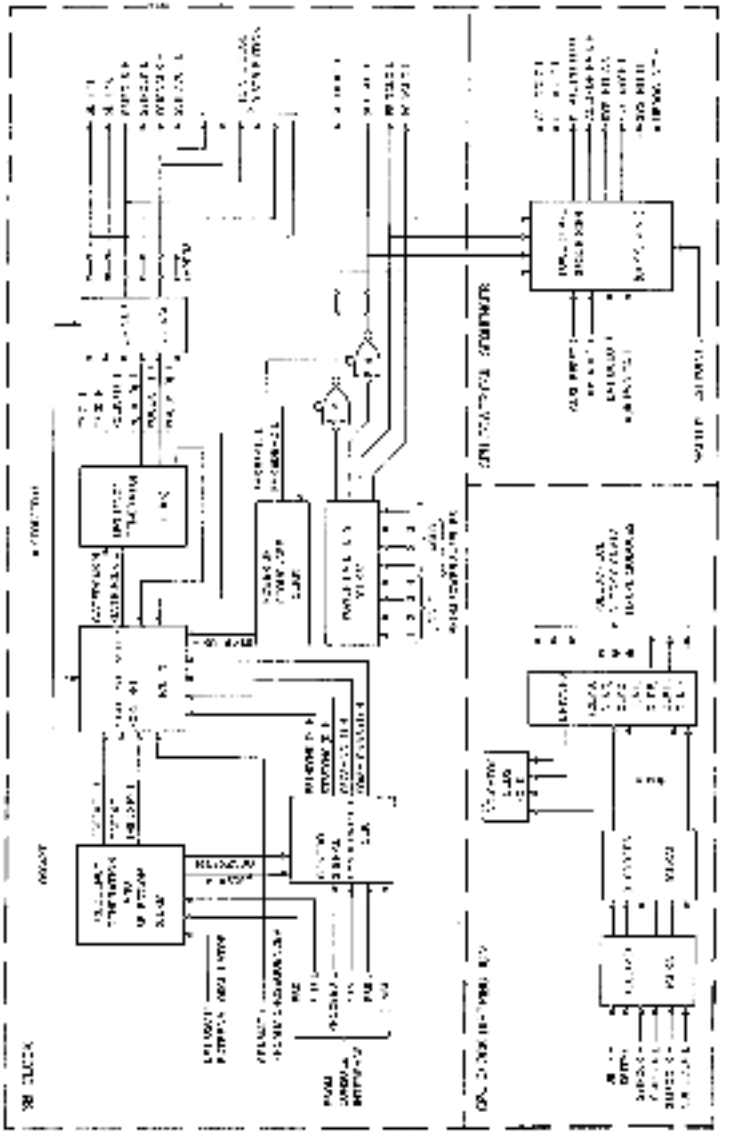
TRANSLATION BUFFER DATA MATRIX BLOCK DIAGRAM



TRANSLATION BUFFER ADDRESS MATRIX BLOCK DIAGRAM



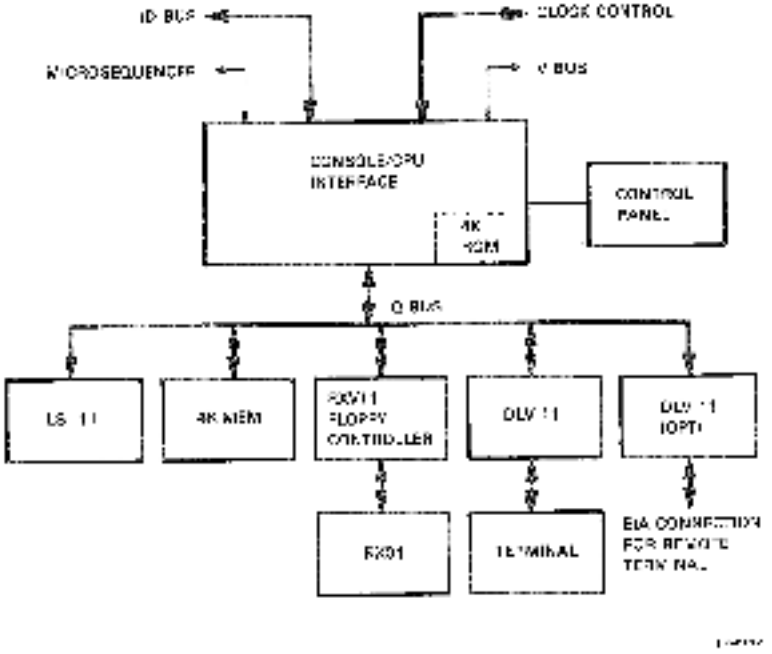
CLOCK MODULE BLOCK DIAGRAM



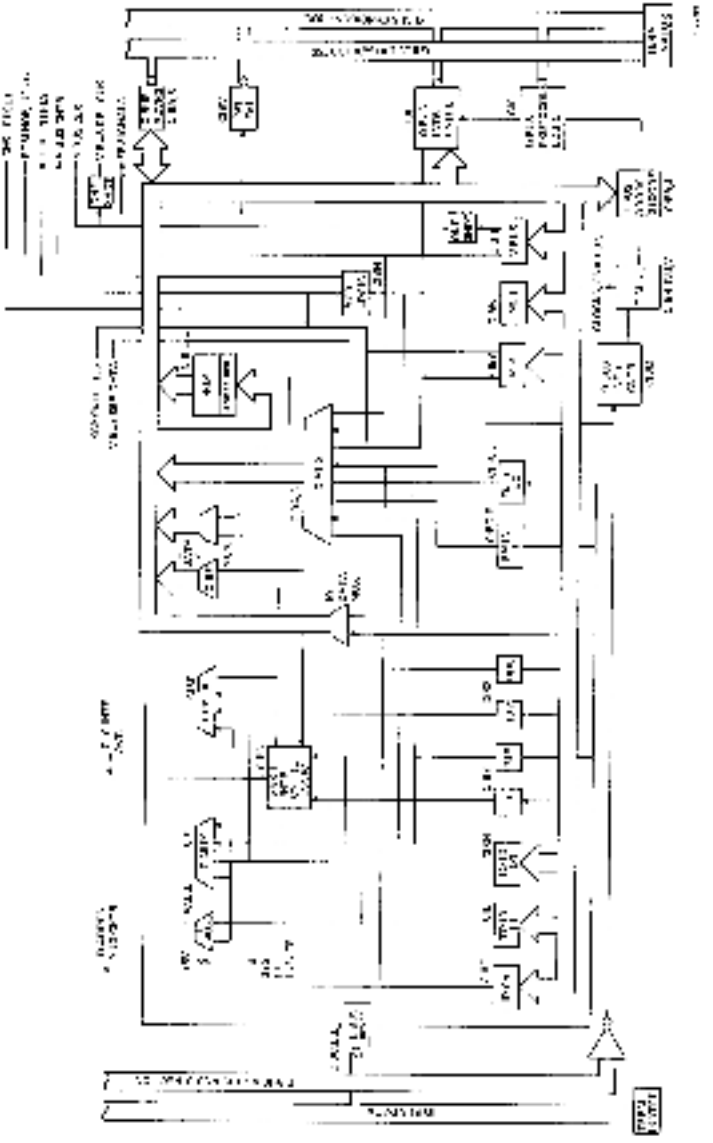
MICROSEQUENCER BLOCK DIAGRAM

FLOATING-POINT ACCELERATOR BLOCK DIAGRAM

CONSOLE SUBSYSTEM CONFIGURATION



CONSOLE INTERFACE BOARD BLOCK DIAGRAM



ID BUS MAP

ID BUS MAP

I/O BUS MAP

ID BUS REGISTER BIT CONFIGURATIONS

ID@: 20	NAME: TDCR
Bit Fields	Description
<31:20>	Data to Communications Buffer Bytes <3:0>
	Read Only Located on KB211 (TEPE)

ID@: 01	NAME: TIME OF DAY
Bit Fields	Description
<31:0>	32 Bit counter 100 Hertz Rate
	Read/Write Located on M3224 (CHRON)

ID@: 03	NAME: SYSTEM ID
Bit Fields	Description
<31:24>	System Type
	01=VAX 11/730
<23:16>	ECO Level
<15:12>	Manufacturing Plant
<11:00>	System Serial Number
	Read Only Silicon by jumpers on Backpanel Used from KB15A CSDC,D,T

ID@: 24	NAME: FWCR
Bit Fields	Description
<07>	Done
	Set by console software signifying data available in RXCR
	Read Only
<06>	Interrupt Enable

ID BUS REGISTER BIT CONFIGURATIONS

	Allows interrupt when Done Bit
	Read/Write
	Located at PB21 (CIOC)
Bit: 15	NAME: RXDB
Bit Fields	Description
<7:0>	Data From Console Subsystem
	Read Only Located at MB21 (CIOC,D)
Bit: 15	NAME: TXCS
Bit Fields	Description
<7>	Ready
	Set by console to indicate Ready to receive data
	Read Only
<6>	Interrupt Enable
	Allow interrupt when Ready set
	Read/Write Located at MB21 (CIOC)
Bit: 15	NAME: TXDB
Bit Fields	Description
<7:0>	Data to console subsystem
	Write Only Located at MB20 (CIOC,D,0)
Bit: 15	NAME: IO
Bit Fields	Description
<15:00>	Bits: Purpose Write: Write Register
	Read/Write

I/O BUS REGISTER BIT CONFIGURATIONS

	Lyout x' ;(: <7:5> R0337 (OCPC) <4:0> R0322: (OOFC) <7:6> R0337 (PCSL,P)
1344: 10	NAME: 2EXC INTERVAL COUNTER
Bit Fields	Description
<7:00>	Only lsbd 'bro interval counter on overflow or XPRR bit in CSR 2040L 886 Wlll only <7:6> R0337 (PCSH) <5:00> R0131 (ICCS)
1040: 10	VAME: INTERVAL CLOCK STATUS
Bit Fields	Description
<15>	Error (A 1 (on ass 'd overflow bad or flag Serviced. Read/Write 1 to Clear
<07>	Interval request Set when counter overflows. Read/Write 1 to Clear
<06>	Interrupt Enable Enables interrupt on rollover read/Write
<05>	Single CLK Advance counter on step Write Only
<04>	XFER Forces next interval to counter Write Only
<03>	RUN

ID BUS REGISTER BIT CONFIGURATIONS

	Allows counter to increment at 1 microsecond rate Read/Write Cleared on MSCLR (INIT)
IDxr ID	**BASE: 172440 + 10*IDR**
Bit Fields	**Description**
<7:00>	32 Bit Q-22 Level 2 All 1 Level-2 word Read Only Cleared on MSCLR (CLEAR) Cleared on MSCLR (INIT)
IDxr ID	**BASE: CPU ERROR STATUS (CES)**
Bit Fields	**Description**
<0>	Address Error: Used by Memory Management Microcode Read Only Cleared on MSCLR (CLEAR)
<1>	Control Store Parity Error Summary "OR" of Control Store Parity Error Bits Read Only Cleared on MSCLR (INIT)
<4:13>	Control Store Parity Error Bits <14>=Group 3 <13>=Group 2 <12>=Group 1 Read Only Cleared on MSCLR (INIT)
<1>	R ALU 3
<2>	R ALU 2
<3>	ALU 3

ID SUS REGISTER BIT CONFIGURATIONS

```
<15>          ALU S

<07>          ALU STT

                   Read/Write
                   Cleared on RESET (LOW)

<6:04>        Arithmetic Trap Code
              1 Decimal Divide by 0
              6=Decimal Overflow
              5=Bits. Underflow
              4=Float Divide by 0
              3=Float Overflow
              2=Integer Divide by 0
              1=Integer Overflow
              0=No Trap Pending

                   Read/Write
                   Located on 8231 (7:04)

<03>          P/C Command Monitor Enable

                   Loaded or Read by Microcode

                   Read/Write
                   Located on 8231 (7:03)

<02:01>       MMU Level

                   Used to Deliver MMU BTB during TRT

                   Read/Write
                   Located on 8231 (7:02)

ID06: ID      NAME: VECTOR

Bit Fields         Description

<15>          Trap Valid

                   Indicates at least one bit was set in the
                   priority field

                   Read Only
                   Located in 82C30 (7:15)

<04:00>       Priority

                   Priority encoded value of bits <7:03> of
                   the word last written into vector register

                   Read Only
```

IO BUS REGISTER BIT CONFIGURATIONS

Location on KB11D (177577)

<15:13>	Number Of Ones
	Number of ones last written into vector register
	Read Only Located on KB11C (177740)
<12:0>	Vector
	Hardware Generated Vector
	Read Only Located on KB11C (177536)

177546 KW11-L/P/AA/BA (177560/177570 KB11C/D/E)

Bit Fields	Description
<15:8>	Interrupt Priority Level Pending
	Level of highest interrupt which is currently active (1-7)
	Read Only Located on KB11D (177546)
<7:6>	Software Interrupt Requests
	Pending Software Interrupt Flags
	Read/Write Located on KB11D (177546)

177560: DECSYSTEM STATUS REGISTER

Bit Fields	Description
<15>	Compatibility Mode
	CPU executing PDP-11 mode instructions
	Read/Write Located on KB11C (177740)
<00>	Trace Enable
	At end of an instruction and if trace pending trap is initiated

	Read/Write Enabled on MBUSI (CCENB)
<21>	Flush Port Done Microcode sets this bit to indicate that certain instructions, during the instruction may be restarted from that point if an interrupt or exception occurs. Read/Write Located on MBUSI (CFHP)
<e5>	Interrupt Mask Disables CPU operating on interrupt mask Read/Write Located on MBUSI (CENP)
<4:2>	Current Mode Current operating mode of System 1=SCR 2=EXCEPTION 3=SUPERVISOR 4=EXECUTIVE 5=KERNEL Read/Write Located on MBUSI (CFHP)
<7:5>	Previous Mode Previous operating mode (before change Mode Name) 0=USER 2=SUPERVISOR 3=EXECUTIVE 5=KERNEL Read/Write Located on MBUSI (CENP)
<0:1>	Interrupt Priority Level Current interrupt priority level of CPU Read/Write Located on MBUSI
<2>	Enable decimal overflow exception
<3>	Enable floating underflow exception

ID BUS REGISTER BIT CONFIGURATIONS

(05)	Enable integer overflow exception Read/Write Located on 68881 (CPU)
(04)	T bit Module is using Trace Pending
(03)	3 bit
(02)	2 bit
(01)	1 bit
(00)	0 bit Read/Write Located on 68851 (MMU)

IDA: 10 — MEMORY MANAGEMENT BUFFER DATA REGISTER

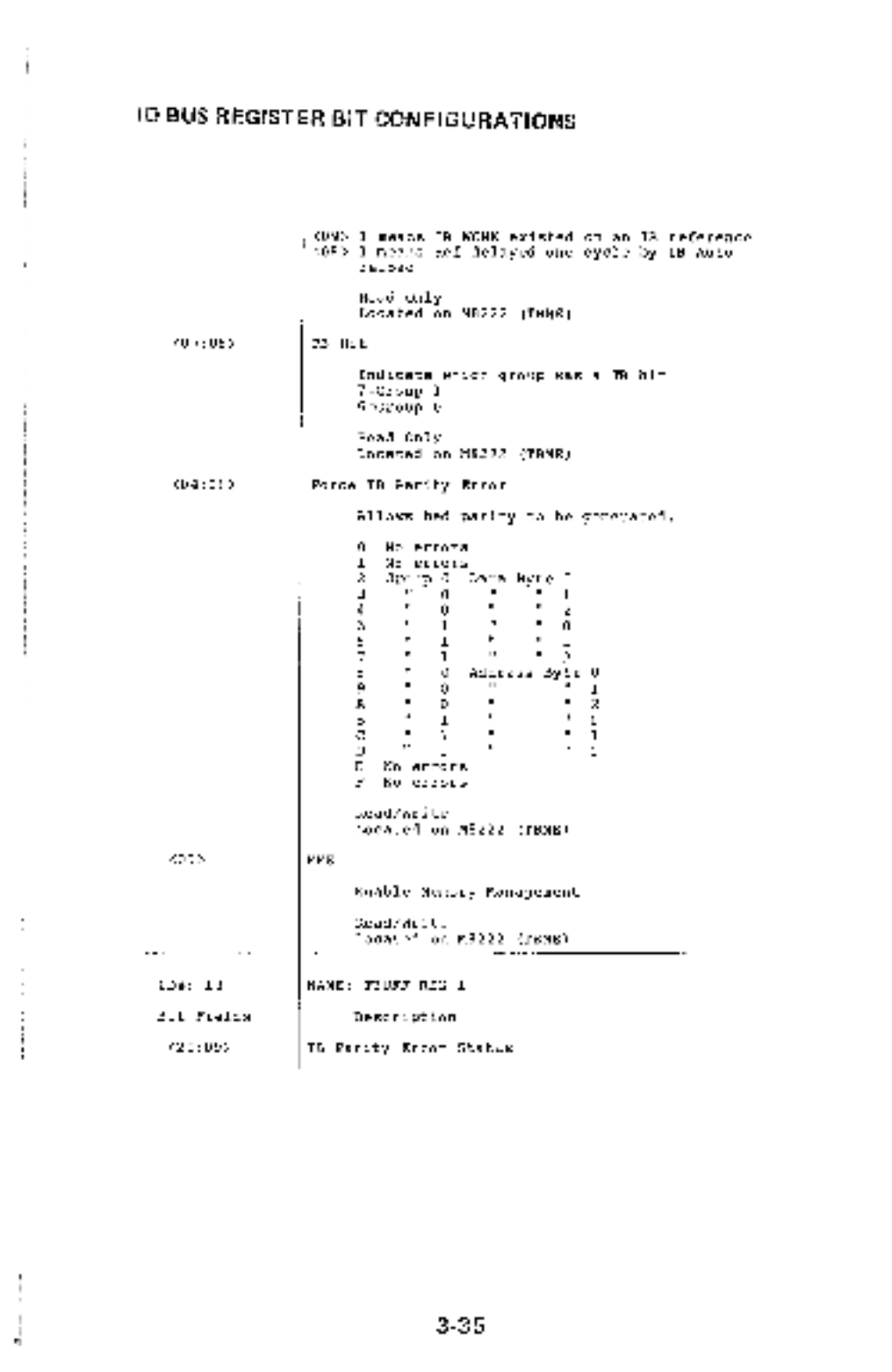
Bit Field	Description
(31)	Valid Allows TS data with UA(1:0)=11 and FC(2:0)=111 and address(31:16) equals UA #(31:16) Write Only Located on MMU (CPU)
(30:27)	Protection Code Allows Protection of Address

	Kernel	Exec	Super	User
0000	*	*	*	*
0001	Unpredictable			
0010	R/W	R	*	*
0011	R	*	*	*
0100	R/W	R/W	R/W	R/W
0101	R/W	R/W	*	*
0110	R/W	RW	*	*
0111	RW	RW	*	*
1000	R/W	R/W	R/W	*
1001	R/W	R/W	R0	*
1010	R/W	RO	R0	*
1011	RO	RO	RO	*
1100	R/W	R/W	R/O	RO
1101	R/W	R/W	RO	RO
1110	R/W	RO	RO	RO

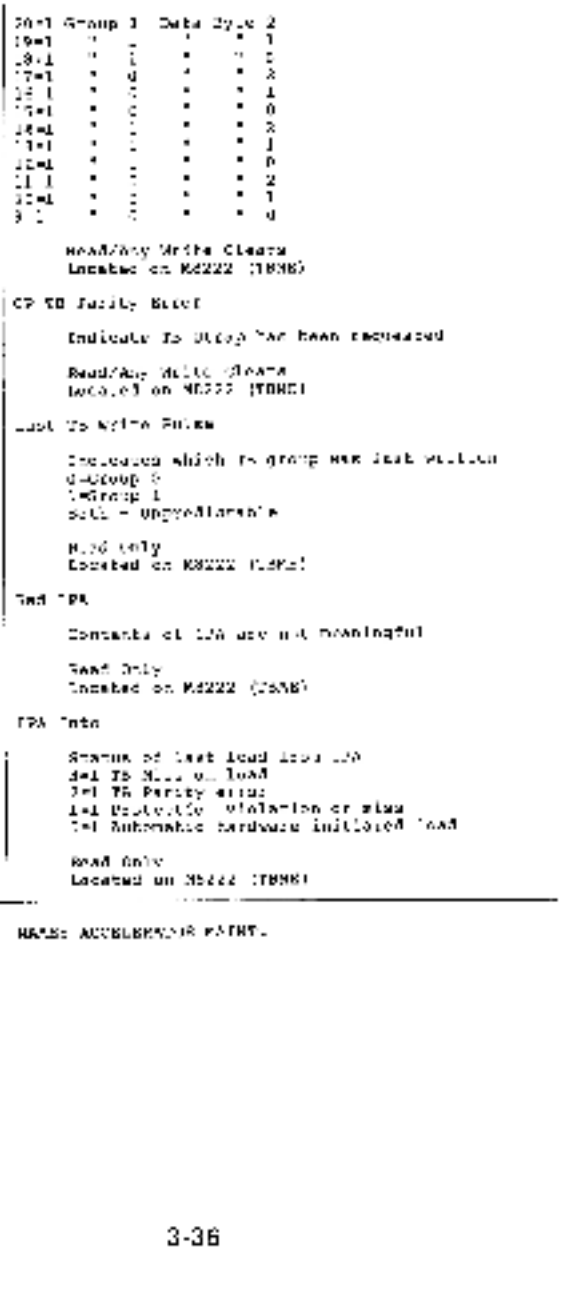
ID BUS REGISTER BIT CONFIGURATIONS

	1'11 R2 x0 R1 R0 X No access R/W Read/Write R0 Read Write Only Located on MB222 (CR97)
<06>	Modify Writes a modifiable pg. Write Only Located on MB210 (CR97)
<10:0>	Page Frame Number When translation occurs these bits bec page number (i.e., PA<21:09> Write Only Located on MB222 (CR98)

Bit: 15	MMAP: 1 BUFF RMG 0
Bit Field	Description
<00:15>	Queue Displace Directs TC writes to C/Cload groups 00=Write DCTU 10=Empty Displace Group 11 Force Replace Group Read/Write Located on MB222 (CR98)
<15:6>	Queue Size Forces TC Miss on C/Cload groups (1=Group 1 16 Group 7 Read/Write Located on MB222 (CR98)
<15:00>	Bus reference Data on Task Mon Bus memory reference <15> Status of MPT file <17> Status of VATC file <15:0> Status of CMCC (level)


(0:0) 1 masks TB NXMR asserted on an IA reference
(0:1) 1 masks self delayed nxm cycle by LB Auto
Read only
Located on MB222 (TBMR)
(0:05)
22 BLK
Enables error group mask & TB bit
7-Group 1
6-Group 0
Read Only
Located on MB222 (TBMR)
(04:01)
Force TB Parity Error
Allows bad parity to be generated.
0 No errors
1 No errors
2 Byte 0 Data Byte 1
3 0 0 * * 1
4 * 0 * * 2
5 * 1 * * 0
6 * 1 * * 3
7 * 0 Address Byte 0
8 * 0 * * 1
9 * 0 * * 2
A * 1 * * 1
B * 1 * * 3
C * * * * 2
D No errors
E No errors
Read/write
Located on MB222 (TBMR)
(03)
PPE
Enable Parity Enhancement
Read/Write
Located on MB222 (TBMR)
LSB: 11
NAME: TBMR REG 1
Bit Fields
Description
(2:00)
TB Parity Error Status

ID BUS REGISTER BIT CONFIGURATIONS



ID BUS REGISTER BIT CONFIGURATIONS

Bit Fields	Description
<31>	Write Trap Address When bit clocks trap address register Write Only Located in 82206 (eSMA)
<30:17>	Trap Address Bit to form 8K4 address on ADT trap Read/Write Located in 82206 (EMIB)
<17>	Write Alarm Match Setting clocks alarm match register from alarm[31:0] of this register Write Only Located in 82207 (RTC)
<16>	Write Rollover Indicates a write which has occurred Read Only Located in 82c37 (DMAC)
<15:00>	Write Break/Current Address Write write back register refer to section Write Protect Mechanism Read/Write Located in 82207 (RTC)

ID: 17	NAME: ACCELERATOR CONTROL (0A108
Bit Fields	Description
<31>	Trap Read/Write Write to this reg will clear Located in 82206 (EMIB)
<30>	Reserved Operand Write zero error

	Read Only Located on MS105 (2MBD)
<15>	Accelerator Enable 1=Enable Accelerator 0=Disable Accelerator Read/Write Located in KB707 (F-07)
<03:00>	Accelerator Type 0=FBA Read Only Located on KE287 (7NFP)

IOB: 10 NAME: SCLO

Bit Fields	Description
	IO location 8170 used to scale EBI activity
<15>	After Read Reset scale after read cleared Write Only Located on KE213 (5+KF)
<14>	SCF Increment Read Only Located on MB211 (5DDJ)
<13:07>	SCL IC<6:0> Read Only Located on KB219 (5+KA)
<06:02>	EBI DAC<2:0> Read Only Located on KE219 (5+KA)
<1:10>	EBI MSR<3:1> or SDT<03:0> Bits written with EBI<07:03> when EBI IO Address received (address, otherwise EBI <03:00> are written Read Only Located on MB219 (5DDJ)

ID BUS REGISTER BIT CONFIGURATIONS

<7:6>	SPC EXP<1:0>
	Read Only Located on MB222 (C8F2)
<5:00>	SPC IR<13:08>
	Read Only Located on MB237 (C9E1)
TCB: 19	NAME: SCSI ERROR REGISTER
Bit Status	Description
<15>	BUS Interrupt Enable
	Enable interrupt for BUS error Read/Write Located at MB213 (BB1A)
<14>	DRQ
	Received corrected (ok) data from memory Ready/write 1 to clear Located on MB214 (BB2C)
<13>	EOP
	Received end data acquisition from memory Ready/write 1 to clear Located on MB210 (C3E7)
<12:11>	CP Timeout Status
	12=1 Timeout for CP sequencer cycle 11 10 0 0 No device response 0 1 Device busy 1 0 Waiting for read data 1 1 Impossible code 10 = Ready/Write 1 to clear Mnemonic Address: 0E, 02 Scsi11 Read Only Located on MB218 (BB0A)
<0>	IP SCSI Reset Confirmation

IB BUS REGISTER BIT CONFIGURATIONS

```
                    Set when CF requested cycle received error
                    confirmation by command ACCCCF ENABLE

                    Read only
                    Write 1 to bit 15 to clear
                    Located on MB1.8 (UBLH)

<07>            15 <05

                    Read Data Substitute for TR Data

                    Read/Write 1 to clear
                    Located on P2218 (SB1F)

<06:04>         IB Timeout STATUS

                    When Timeout for TR requested cycle

                    05  04
                    0   0     No device response
                    0   1     Device busy
                    1   0     Waiting for read data
                    1   1     Impossible code

                    6 = Read/Write 1 to clear
                    Also clears L[1:0]<5:3>
                    05:04 = Read Only
                    Located on P2218 (SA1F)

<03>            IB CDI Error Confirmation

                    Set when IB requested cycle received error
                    confirmation

                    Read only
                    Write 1 to bit 6 to clear
                    Located on P2218 (SA1F)

<02>            Multiple CF error

                    Set with pending CF timeout or CF SBI error
                    confirmation not received

                    Read only
                    Write 1 to bit 13 to clear
                    Located on MB1.8 (UBLH)

<1>             RST Not Busy

                    Read Only
                    Located in P2218 (SA1F)
```

ID BUS REGISTER BIT CONFIGURATIONS

ID(7:0)	NAME: PHYSICAL ADDRESS
Bit Fields	Description
	Contains Physical Address on BI (Arbitration will not take place if ID Does Change) Read Only Latched until CPU Timeout Reset by [illegible]
<31:30>	Mode 31 30 0 0 Kernel 0 1 Executive 1 0 Supervisor 1 1 User Implemented as M1219 (BI3)
<29>	Protection Check Equal 1 for references to subset of hardware pro ection check Implemented as M1219 (BI4)
<28:03>	Physical Address <28:03> Bus<29:03> Latched on <27:16> (BBNN,I) <15-03> (BB,E,H)

ID(7:0)	NAME: BI Fault Register Contents
bit Fields	Description
<31>	Parity Error BI Parity Fault Read Only Latched on M3111 (BBNN)
<29>	Unexpected Read Data Fault. Read Only Latched on M3219 (BBH)
<27>	Multiple Transmitter Fault Read Only Latched on M3219 (BBE)

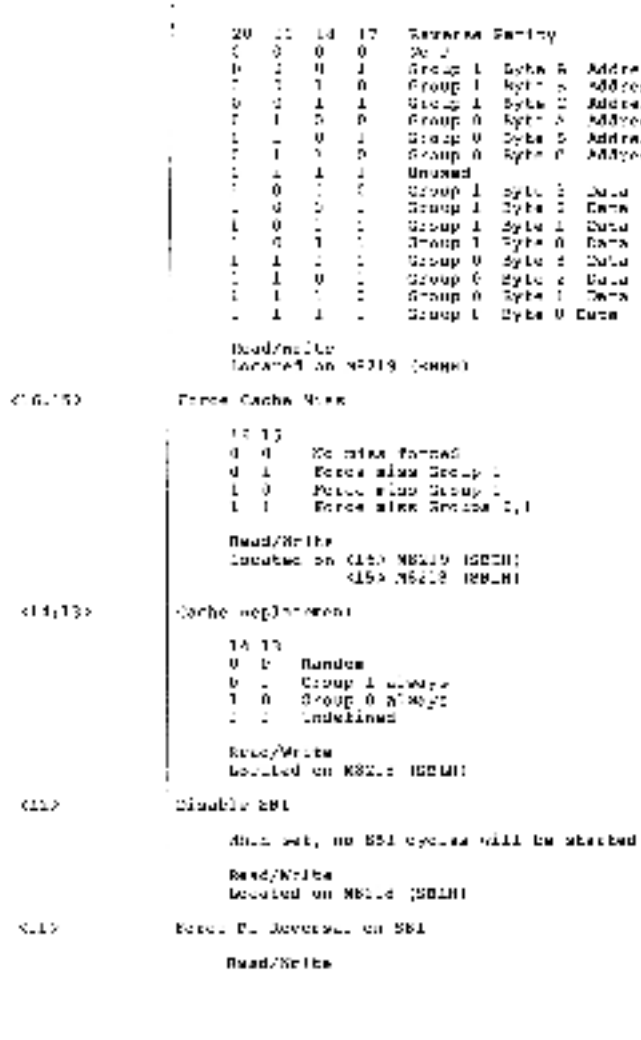
IO BUS REGISTER BIT CONFIGURATIONS

ID BUS REGISTER BIT CONFIGURATIONS

```
<11>        Dual Bus Lock

            A.   Lock Unconditional. (See Bit 10)
                 Force when Count (Bits 15:8)=0

            B.   Conditional Lock
                 Lock  when  partial  conditions  exist,
                 C/J_status looks at SSI, Mbus match, compare
                 signal is generated which allows counter to
                 decrement.

                 Bus counts =0, Bibi will lock

                 Unlock by writing number =0 into counter

                 Read Only
                 Clear by writing number not equal 0 to
                 counter
                 Located in P3019 (Bus)

<10>        SSI Lock Interrupt Enable

                 Read/Write
                 Located in P3019 (SSI)

<9>         Lock  unconditional

                 Enables Bibi lock Bus counter  1

                 Read/Write
                 Located on MB019 (MBus)

<8:7>       Conditional Lock Codes

                 Bit #7
                 0  0   No Compare
                 0  1   Eq, Only
                 1  0   GE, Only
                 1  1   Diag, Command Function or Mask

                 Read/Write
                 Located on P3019 (Bus)

<7:3>       Compare Command or Mask<3:0>

                 Read/Write
                 Located on P3019 (SSI) (MBus)

<2:1>       Function Te[1:0]

                 Read/Write
                 Located on MB11 (MBus)

<0:6>       Count Value<1:0>
```

	Khan =F allows bit lock Read/Write Located on X8213 (8888)
IO<n:0>	HOST PATHTRENCE REGISTER
Bit Field	Description
<31>	Force PE Reversal on EBI Read/Write Located on X8213 (8883)
<30>	Force Write Sequence Fault Read/Write Located on X8213 (8882)
<29>	Force Bus/Internal Read Data Fault Causes (cancels if SET TAG=0, Maintenance Op, Undefined Data, good parity for unexpected read Data if a matches means Read/Write Located on X8219 (2222)
<28:25>	Maintenance TAG<3:0> Used to force unexpected read data faults Read/Write Located on X8218 (2882) (8888)
<24>	Force EBI Invalidate Forces Write data by EBI on EBI to become cache invalidate Read/Write Located on X8213 (8888)
<23>	Enable EBI Invalidate Allows EBI writes to invalidate cache Must be 1 for normal operation Read/Write Located on X8219 (8288)
<22:16>	Reserved Cache Parity

IO BUS REGISTER BIT CONFIGURATIONS



ID BUS REGISTER BIT CONFIGURATIONS

Located on M8218 (8H-1K)

<10:09> Cache Match

11-1 Group 1 Cache match
00-1 Group 0 Cache match

Read Only
Located on M8219 (8-1K)

<08> Cache Timeout

Forces load timeout.

Read/Write
Located on M8219 (8-1K)

104: 16 NAME: CP/IP PARITY ERROR REGISTER

Bit Number	Description
<15>	Any Error Set when cache parity error on CP or ID cache operations Read/Write 1 clears enables register Located on M8218 (8B-1K)
<14>	CP Error When bits 15 and 14 are set, CP reference caused error When bit 15 is set and bit 14 is clear, the CP reference caused no error Read Only Located on M8218 (8F-1K)
<13:06>	Data Parity C.W. If Par Parity D.F., bit is used by ... for ...

13	Parity	07	06	Group	1	Byte 0
14	*	*	*	*	1	*
13	*	*	*	*	1	*
12	*	*	*	*	1	*
11	*	*	*	*	1	*
10	*	*	*	*	0	*
9	*	*	*	*	0	*
8	*	*	*	*	0	*
7	*	*	*	*	0	*
6	*	*	*	*	0	*

ID BUS REGISTER BIT CONFIGURATIONS

	Read Only Located on 82218 (11:0) (SD15) (7:0) (SD7:0)
(Port 20)	Address Parity (A/P) If Bus Parity C.K., bit 15 must be set for meaningful info. 7 Parity CK COM SD_up 2 Byte 2 6 * * * * 0 * 1 5 * * * * 0 * 2 4 * * * * 1 * 0 3 * * * * 1 * 1 2 * * * * 1 * 2 Read Only Located on 82215 (SDLP)
ID#: 21	DATA: 00-FF
Bit Fields	Description
(7:0)	Reading pops bus address into receive stack Writing pushes address on bits 7 ack (Port 2) - (on_to) Bits: Address(7:0)
	Read/Write Located on PB205 (CTCD)
ID#: 22	PARE: 00-FF
Bit Fields	Description
(7:0)	Turns users to compare their PC bus accps with or accepting system clock with ECHO out
	Read/Write (Port 2 - 3) 82139 (Port)
ID#: 23	MASK: Mic Allowed
Bit Fields	Description
(7)	Invert Parity User can invert MIC parity Read/Write Located on MB215 (USCT)

ID BUS REGISTER BIT CONFIGURATIONS

(15:12) Module 2 Count[3:0]

Counter used to point to which of the quantity of WCS is to be written.

R/W/Hyl 0
Located on MB105 (U50CD)

(11:00) Control Store Address

Use to Address WCS for write

Read/Write
Located on MB105 (U50CD)

ID#: 01 MBADs WCS Ctrl

Bit Fields Description

(15:00) Data
Used to write data into WCS

(11:00) Number of WCS Module present

3 2 1 0	WCS	Present
1=	1-15	*
2=	2-14	*
3=	3-13	*
4=	4-12	*
5	5-11	*
6=	6-10	*
7=	7-09	*

(15:12)Write Only
(11:0)Dual/Multi
Located on MB211 (M30B)

ID BUS REGISTER BIT CONFIGURATIONS

Dec	NAME	
31	PICR	All Registers <31:0>
30	PISA	
29	ISR	<31:24> PE3:0> CEIR
		<23:16> A3:0> CBIR
28	KTP	<15:08> A3:0> ICLR
27	EOP	<07:00> A3:0> ICLP
26	SCP	
25	UDP	
24	ILP	ID Register 24 thru 2:
23	PIWR	Are stored in A range
22	D.SY	on CBUS, ICL
21	D.ES	
20	TP	ID thru 35
19	**	Are stored in B range
18	TX	on CBUS, ICL
17	**	
16	T4	
15	T5	
14	T6	All registers
13	**	Read/Write
12	T8	
11	T9	
10	PCSD	
09	SCSD	
08	PTIN	
07	ILLM	
06	SIR	

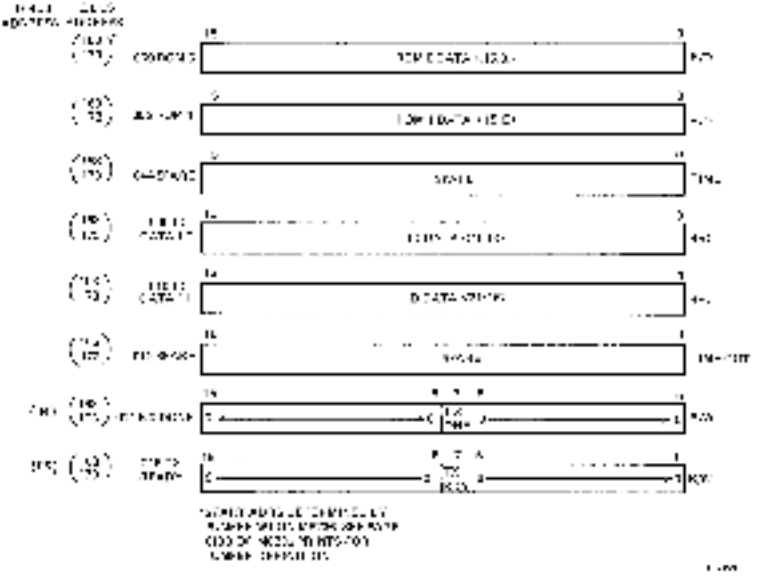
I/O Transfer Control Signals

Name	Description
BSYNC L	Synchronize – The bus master (LSI-11 processor) asserts BSYNC L to indicate that it has placed an address on BDAL (17:00) L. The transfer is in progress until BSYNC L is negated.
BDIN L	Data Input – The LSI-11 asserts BDIN L for two types of operations: 1. When it is asserted during BSYNC L time, BDIN L specifies an input transfer with regard to the processor. It requires BRPLY L as a response. The processor asserts BDIN L when it is ready to accept data from the slave device. 2. When the processor asserts BDIN L without BSYNC L, it is requesting an interrupt vector from an interrupting device.
BDOUT L	Data Output – When the LSI-11 processor asserts BDOUT L, valid data is or can be on an output transfer from the processor to an I/O slave device. The slave cannot destroy BDOUT L (guards) before storing the data. The slave device responding to the BDOUT L signal must assert BRPLY L to complete the transfer.

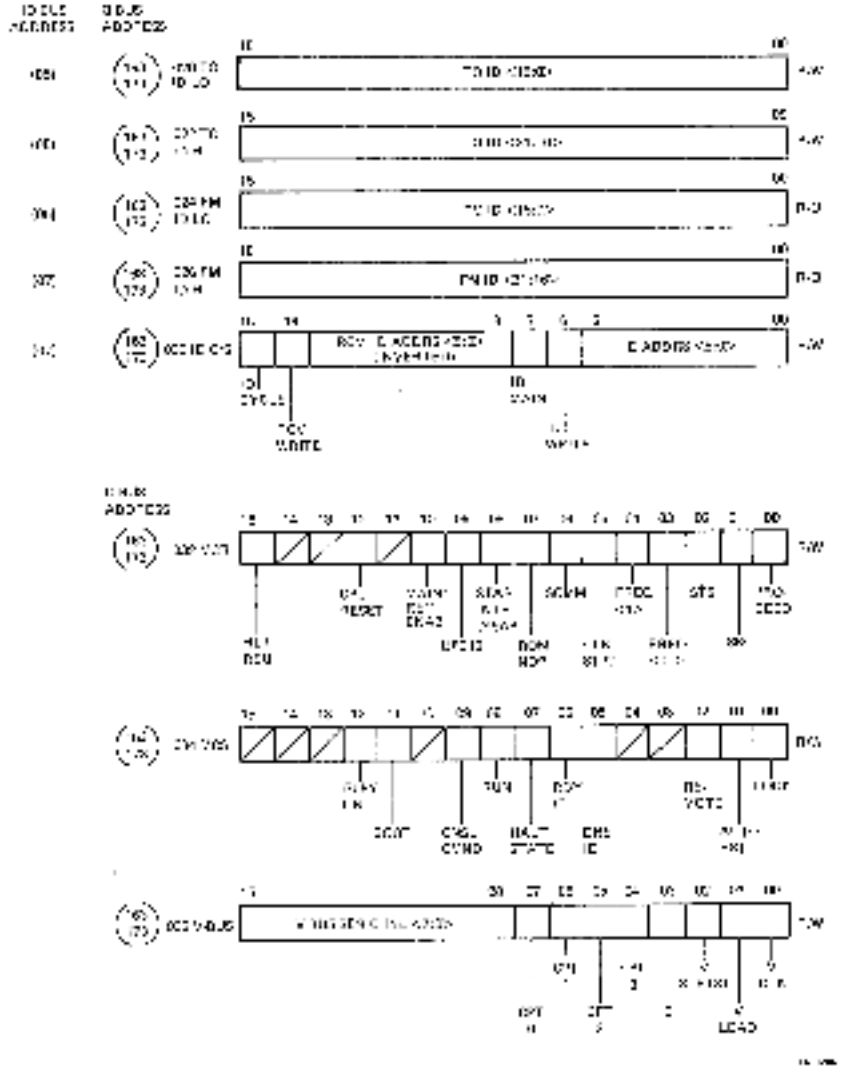
Name	Description
I/O Transfer Control Signals	
BWTBT L	Word/Byte – The LSI-11 processor uses BWTBT L to control bus cycles in two ways: 1. The processor asserts BWTBT L on the leading edge of BSYNC L to indicate that an output sequence (DATOB or DATOBU) is to follow. 2. The processor asserts BWTBT L together with BDOUT L on a DATOB cycle for byte addressing.
BRPLY L	Reply – A slave device asserts BRPLY L in response to BDIN L and BDOUT L on data transfers and in response to BIAKO L during interrupt transfers. BRPLY L indicates that the slave has asserted input data on the bus, accepted output data from the bus, or asserted an interrupt vector on the bus.
Interrupt Control Signals	
BIRQ L	Interrupt Request – A device asserts this signal when its interrupt enable and interrupt request flip-flops are set. BIRQ L informs the processor that a device has data to send to the processor (input device) or is ready to accept output data from the processor. If the processor's PS word bit 7 is 0, the processor responds by acknowledging the request, asserting BIAKO L and BIAKI L.
BIAKO L and BIAKI L	Interrupt Acknowledge Output and Interrupt Acknowledge Input – The processor uses this signal in response to an interrupt request (BIRQ L). The processor asserts BIAKO L which is routed to the O bus on the BIAKI L pin of the first device on the bus. If this device is requesting an interrupt (asserting BIRQ L), it will block the passing of BIAKO L to the next device and then place the interrupt vector on the bus. At the same time the device will negate BIRQ L and assert BRPLY L. If the device is not asserting BIRQ L, it passes BIAKI L to the next device via its own BIAKI L pin and the BIAKI L pin of the next priority device.
Address and Data Signals	
BDAL <15:00> L	These 16 lines form the data/address path. Address information is first placed on the bus by the bus master (processor). The processor then either receives input data from or transmits output data to the addressed slave device or memory location over the same 16 bus lines.
BBS7 L	Bank 7 Select – The bus master asserts BBS7 L when an address in the upper 4K word (address in the 28K–32K range) is placed on the bus. BSYNC L is then asserted, and BBS7 L remains active for the duration of the addressing portion of the bus cycle.

Name	Description
	Initialization, Power Fail Signals
SPOK H	Power OK – The power supply asserts this signal when primary power is normal. If BDOK H is negated during processor operation, the processor initiates a power fail trap sequence.
BDCOK H	DC Power OK – The power supply asserts this signal when there is sufficient dc voltage available for system reliable system operation.
BINIT L	Initialize – The processor asserts BINIT L to initialize or clear all devices connected to the Q bus. The signal is generated in response to a power up condition (the negated condition of BDCOK H).
	Halt and Refresh Signals
BHALT L	Processor Halt – When BHALT L is asserted, the processor responds by halting normal program execution. Memory interrupts are ignored, but memory refresh interrupts are executed if M9312 is the processor module is removed. When the processor is in the halt state, it executes the ODT microprocessor, invoking console service (terminal) operation.
BREF L	Memory Refresh – This signal can be asserted by a processor or console-generated refresh interrupt sequence (when enabled) or by a Q bus dual device. BREF L forces all dynamic MOS memory arrays to be refreshed for each BSYNC L/BDIN L bus transaction.

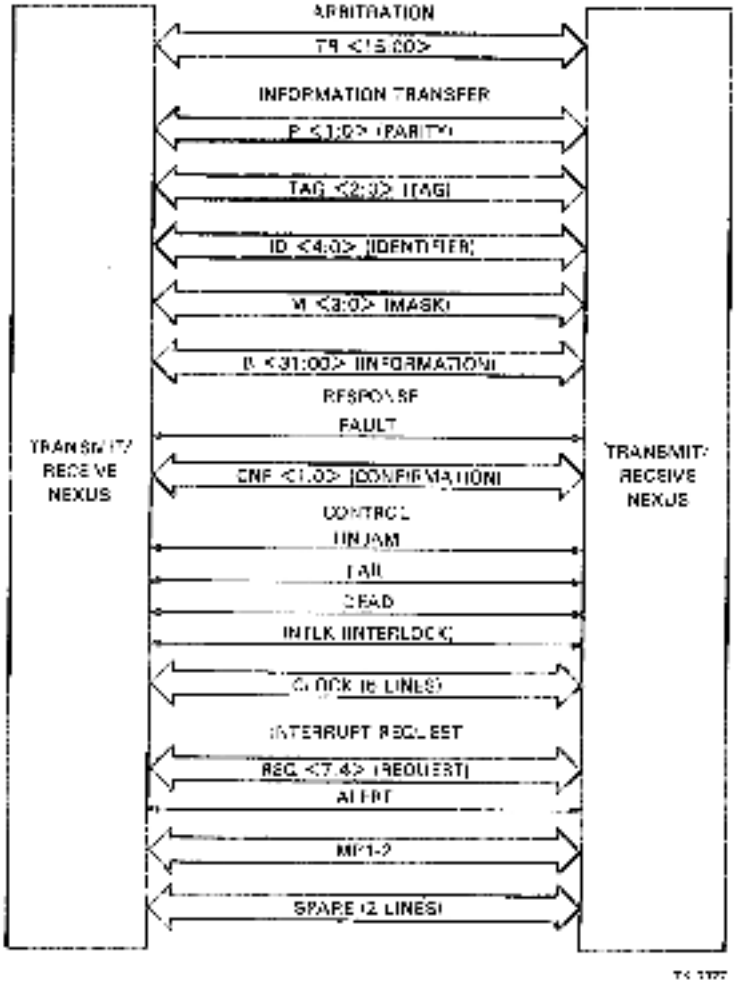
CIB O BUS REGISTERS (LOWER)



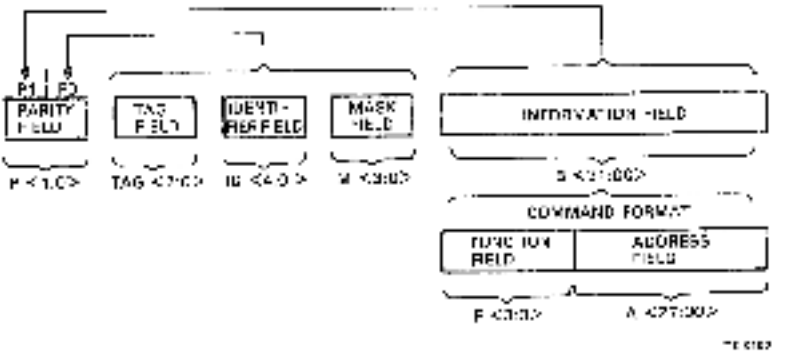
CIB D BUS REGISTERS (UPPER)



SBI CONFIGURATION



SBI PARITY FIELD CONFIGURATION

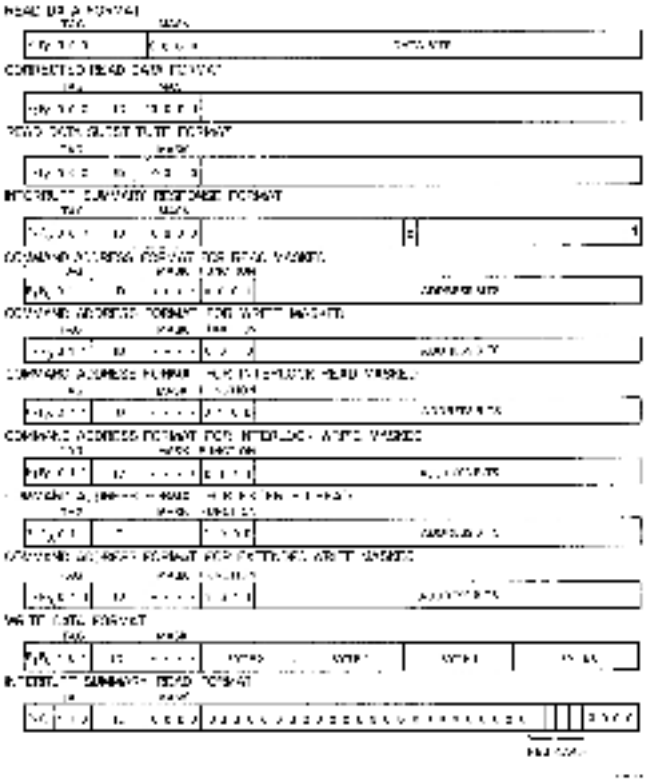


Field	Description
Arbitration Group	
Arbitration Field (ID<15:28>)	Establishes a fixed priority among nexus for access to and control of the information transfer path.
Information Transfer Group	
Information Field (I<31:0>)	Bi-directional lines that transfer data, commands/address, and interrupt information between nexus.
Mask Field (M<3:0>)	Primary function encoded to indicate a particular byte within the 32-bit information field (I<31:0>).
	Secondary function in conjunction with the tag field, indicates a particular type of nexus data.
Identifier Field (ID<4:0>)	Identifies the logical source or destination of information contained in I<31:0>.
Tag Field (TAG<2:0>)	Defines the transmit or receive information types and the interpretation of the content of the ID and information fields.
Function Field (F<3:0>)	Specifies the command code in conjunction with the tag field. This field is part of the 32 bit information field.
Parity Field (P<3:0>)	Provides even parity for CI information transfer path fields.
Response Group	
Confirmation Field (CNF<2:0>)	Provides for a matching nexus to specify one of four response types and indicate its readiness to respond to the transmitter's request.
Fault Field (FAULT)	A cumulative error in the CPU Cat indicates one of several errors stored in the transmitting nexus fault register and the associated SBI cycle in which the error occurred.

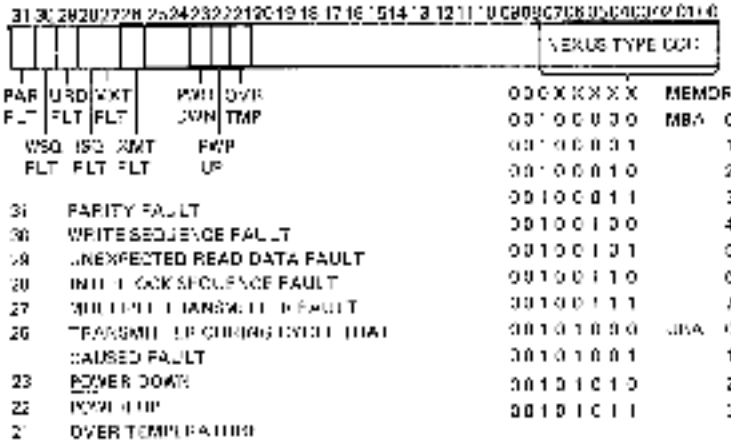
SBI FIELD DESCRIPTION

Field	Description
Interrupt Request Group	
Request Field (REQ (3:0))	Allows a node to request an interrupt to service a condition requiring CPU intervention. Each request line represents a level of node request priority.
Alert Field (ALERT)	A common status line that allows those nodes not equipped with an interrupt mechanism to indicate a change in power or operating conditions.
Control Group	
Clock Field (CLOCK)	Six control lines that provide the clock signals necessary to synchronize SBI activity.
Init Field (INIT)	A single line from the restart node to provide a restart signal to the CPU for initiating system restart operations.
Dead Field (DEAD)	A single line to the CPU to indicate an impending clock circuit or SBI terminating network power failure.
Unjam Field (UNJAM)	A single line from the CPU to the restart node that initiates a restart operation.
Interlock Field (INTLK)	A single line that provides coordination among nodes responding to certain read/write commands to ensure exclusive access to shared data structures.

SBI INFORMATION TRANSFER FORMATS

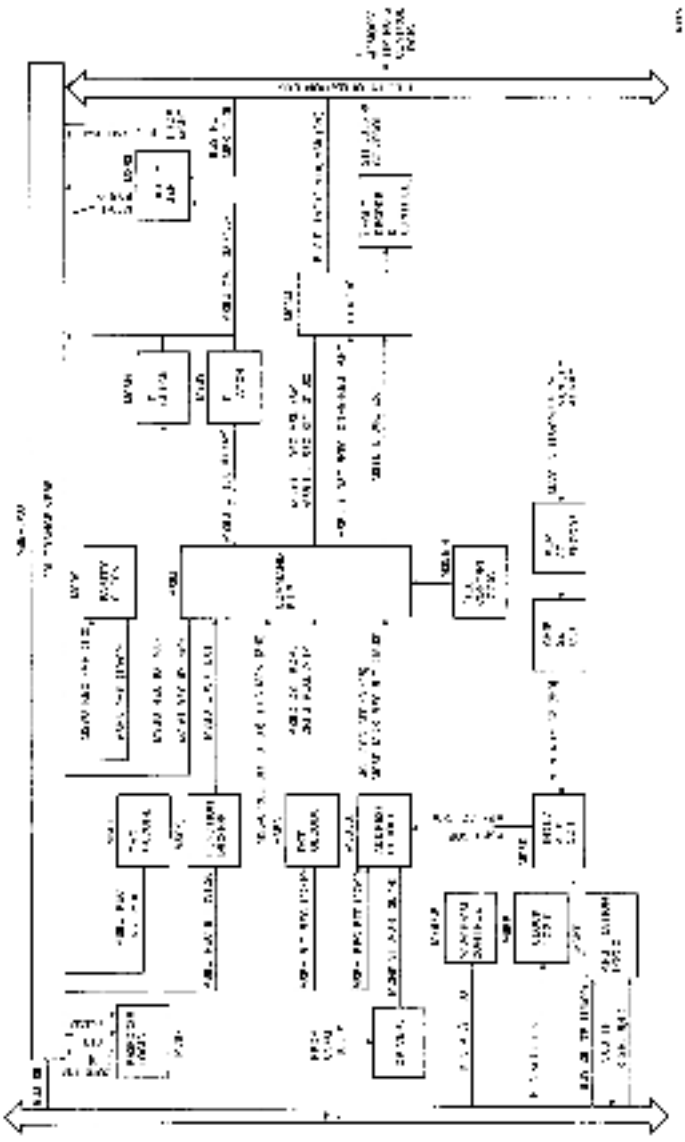


SBI FAULTS

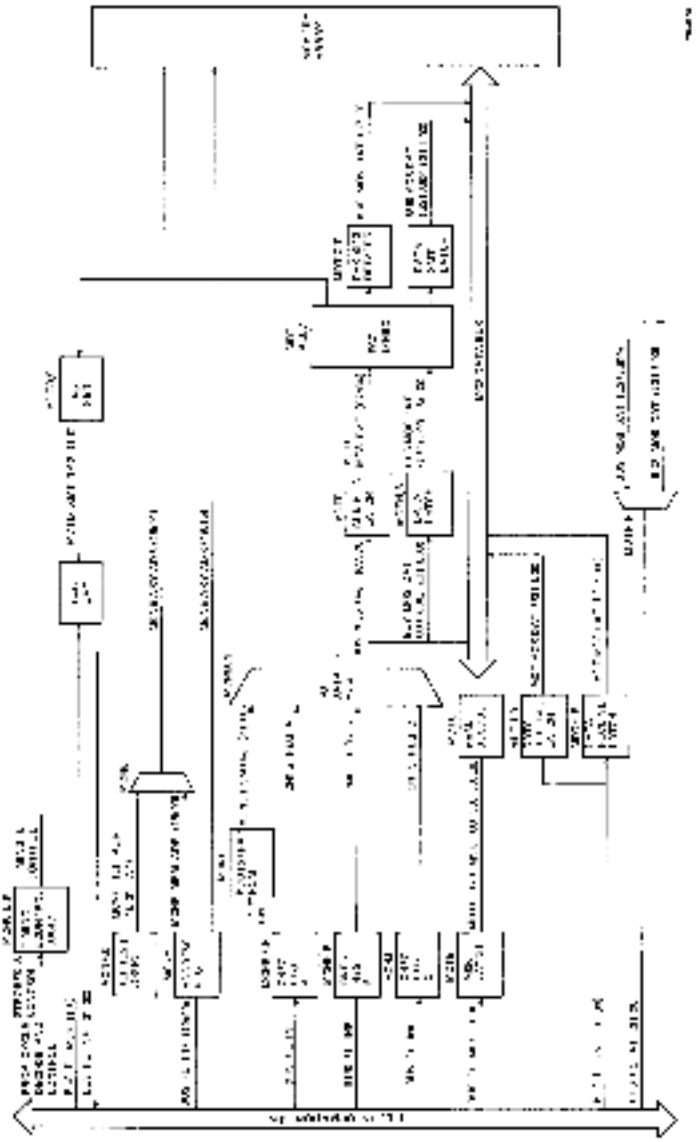

VERUS TYPE CODE
PAR FLT
URD FLT
WKT FLT
PWR DWN
OVR TMP
WSQ FLT
ISQ FLT
XMT FLT
PWR UP
0 0 0 X X X X X
0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 1
0 0 1 0 0 0 1 0
0 0 1 0 0 0 1 1
0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 1
0 0 1 0 0 1 1 0
0 0 1 0 0 1 1 1
0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 1
0 0 1 0 1 0 1 0
0 0 1 0 1 0 1 1
MEMORY MBA 0
1
2
3
4
5
6
7
MBA 0
1
2
3
31 PARITY FAULT
30 WRITE SEQUENCE FAULT
29 UNEXPECTED READ DATA FAULT
28 INTERLOCK SEQUENCE FAULT
27 MULTIPLE TRANSMIT ERROR FAULT
26 TRANSMIT TIMEOUT FAULT
CAUSED FAULT
23 POWER DOWN
22 POWER UP
21 OVER TEMPERATURE

SBI SIGNALS, BACKPLANE PINS

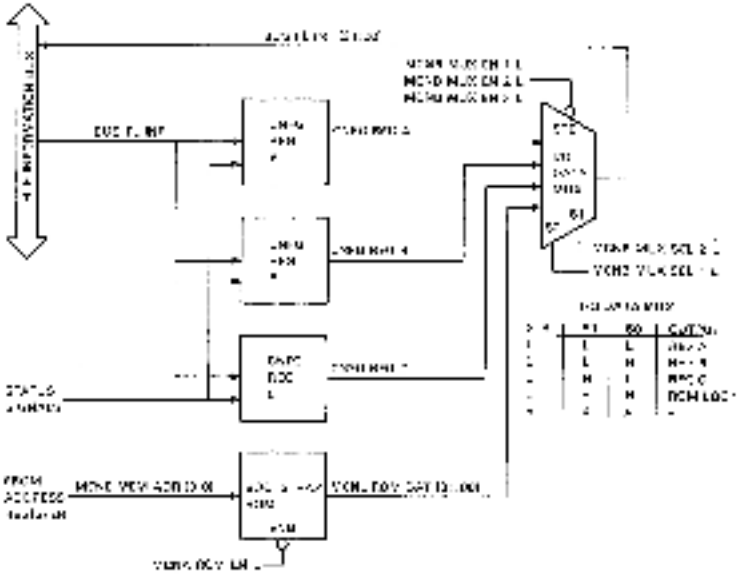
MEMORY BLOCK DIAGRAM, PART 1



MEMORY BLOCK DIAGRAM, PART 2



MEMORY I/O DATA LOGIC



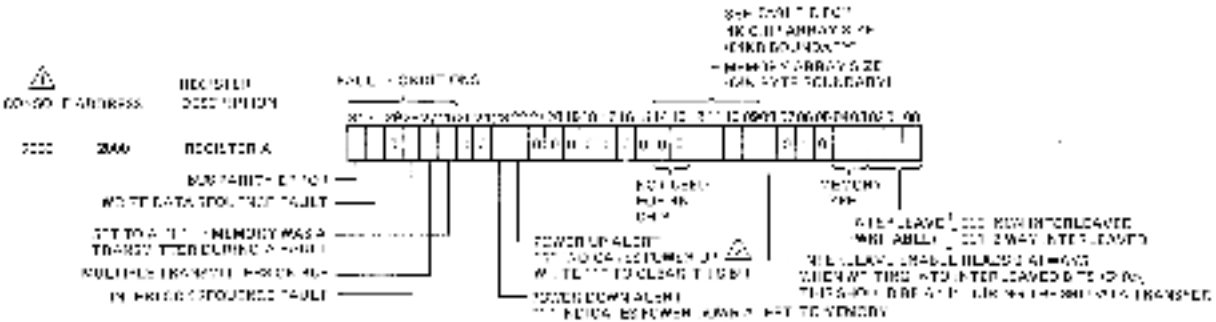


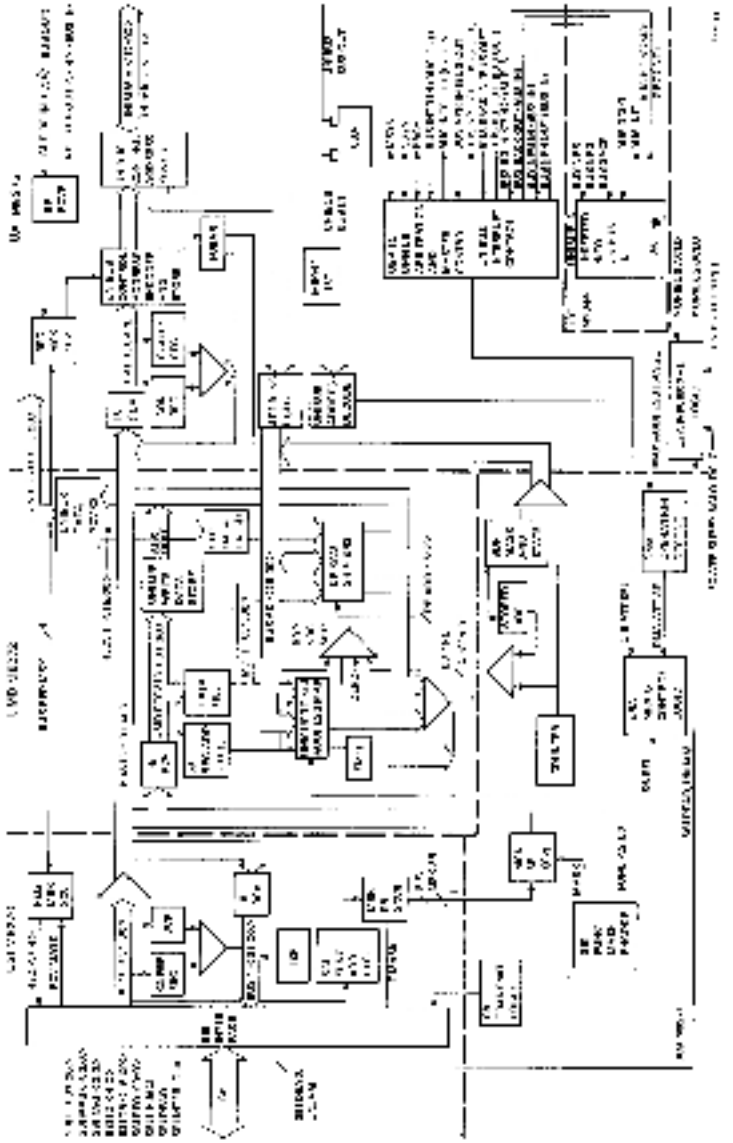
TABLE A 4K CHIP

15	14	13	12	MEMORY ARRAY SIZE
0	0	0	0	64K BYTE
0	0	0	1	128K BYTE
0	0	1	0	256K BYTE
0	0	1	1	512K BYTE
0	1	0	0	576K BYTE
0	1	0	1	704K BYTE
0	1	1	0	832K BYTE
0	1	1	1	960K BYTE
1	0	0	0	1088K BYTE
1	0	0	1	1216K BYTE
1	0	1	0	1344K BYTE
1	0	1	1	1472K BYTE
1	1	0	0	1600K BYTE
1	1	0	1	1728K BYTE
1	1	1	0	1856K BYTE
1	1	1	1	1984K BYTE

TABLE B 64K CHIP

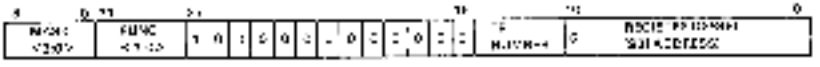
15	14	13	12	11	10	MEMORY ARRAY SIZE
0	0	0	0	–	–	128K BYTE
0	0	0	1	–	–	256K BYTE
0	0	1	0	–	–	384K BYTE
0	0	1	1	–	–	512K BYTE
0	1	0	0	–	–	640K BYTE
0	1	0	1	–	–	768K BYTE
0	1	1	0	–	–	896K BYTE
0	1	1	1	–	–	1024K BYTE
1	0	0	0	–	–	1152K BYTE
1	0	0	1	–	–	1280K BYTE
1	0	1	0	–	–	1408K BYTE
1	0	1	1	–	–	1536K BYTE
1	1	0	0	–	–	1664K BYTE
1	1	0	1	–	–	1792K BYTE
1	1	1	0	–	–	1920K BYTE
1	1	1	1	–	–	2048K BYTE

UBA (DW780) BLOCK DIAGRAM



UBA ADDRESS SPACE AND C/A FORMAT

SBI C/A Format for UBA Register Access



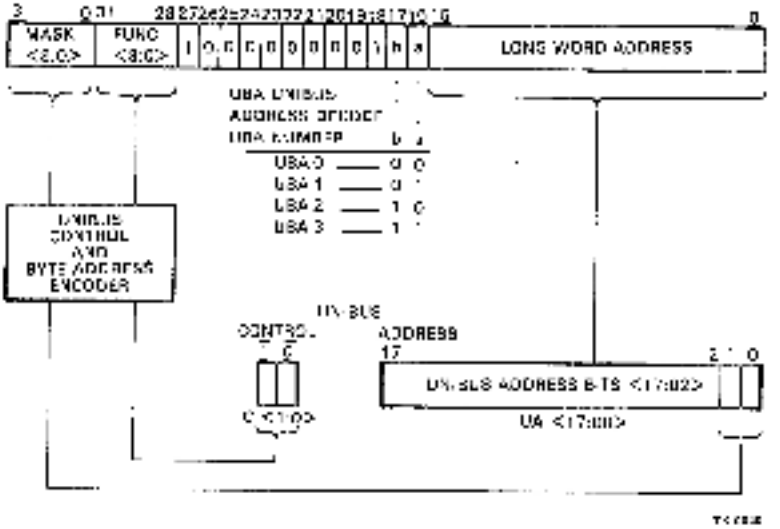
TR Num (Base 10)	Base Address (Physical hex)	SBI Address (hex)	TR Num (Base 10)	Base Address (Physical hex)	SBI Address (hex)
			8		
	2000C000		9		
2			10		
3			11		
4			12		
5			13		
6			14		
7	2000E000		15		

Register Offsets

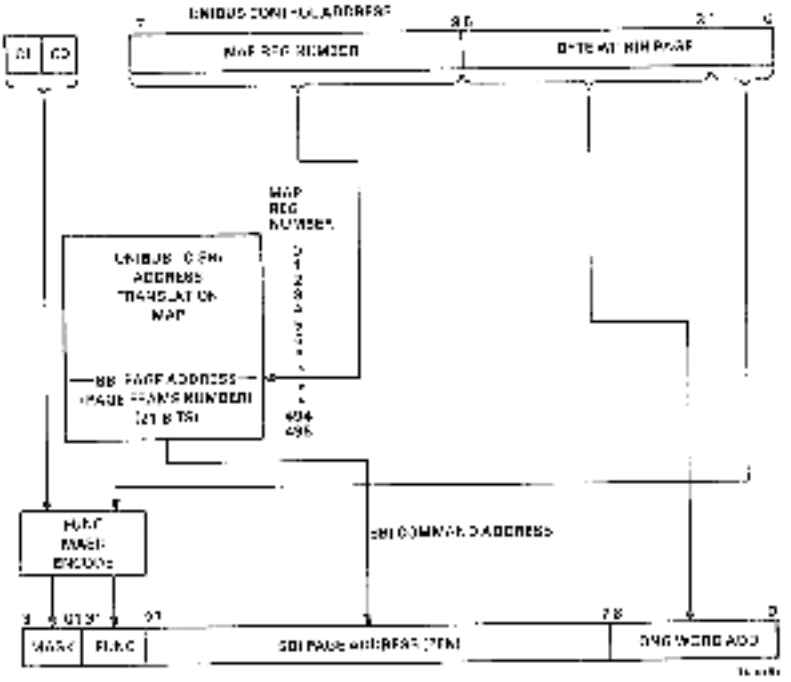
UBA Reg	Base Address (Physical hex)	SBI Address (hex)	UBA Reg	Byte Address (Physical hex)	SBI Address (hex)
CNFGR	000				
UBACR	004				
UBASR	008				
DCR	00C				
FXCER	010		Reserved		
FUBAR	014				
FNER	018				
FUBAR	01C		Reserved		
BRSVR 0	020		WBR 0		
BRSVR 1	024		WBR 1		
BRSVR 2	028				
BRSVR 3	02C				
BRRVR 4	030		WBR #04		
BRRVR 5	034		WBR #05		
BRRVR 6	038		Reserved		
BRRVR 7	03C				
DPR 0	040		Reserved		
DPR 1	044				

SBI TO UNIBUS CONTROL ADDRESS TRANSLATION

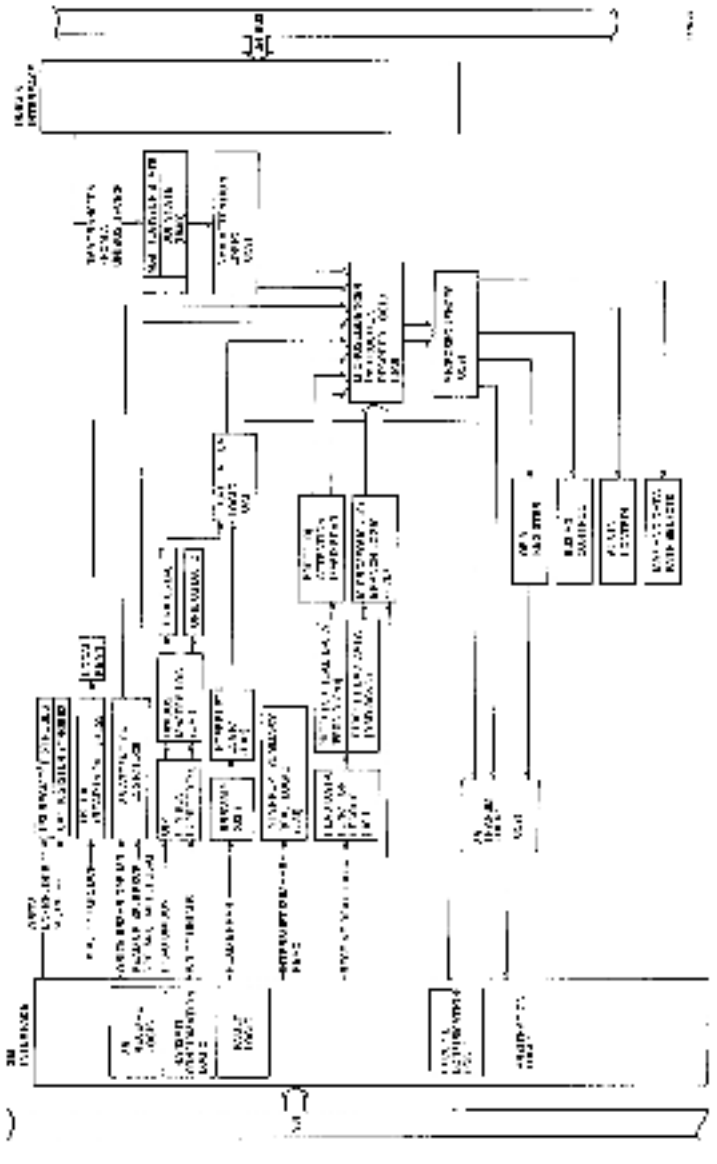
SBI COMMAND ADDRESS FORMAT



UNIBUS TO SBI ADDRESS TRANSLATION

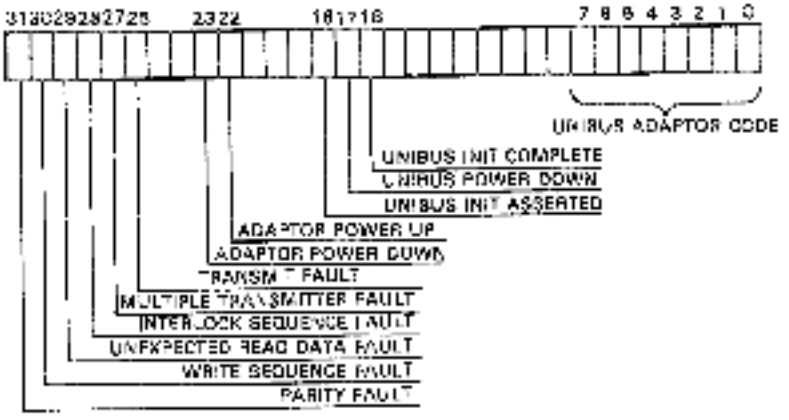


SIMPLIFIED FLOW OF MAJOR CONTROL FUNCTIONS WITHIN THE USA

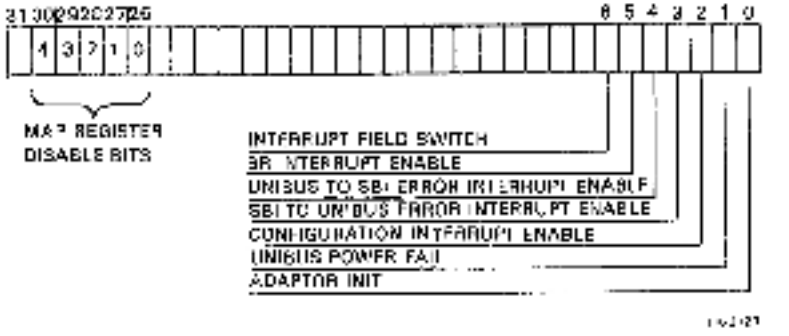


UBA REGISTERS

UBA CONFIGURATION REGISTER, BIT CONFIGURATION

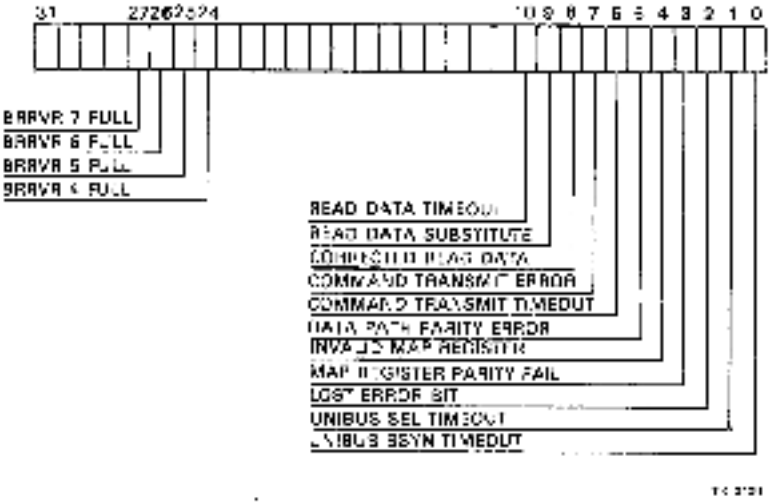


UBA CONTROL REGISTER, BIT CONFIGURATION

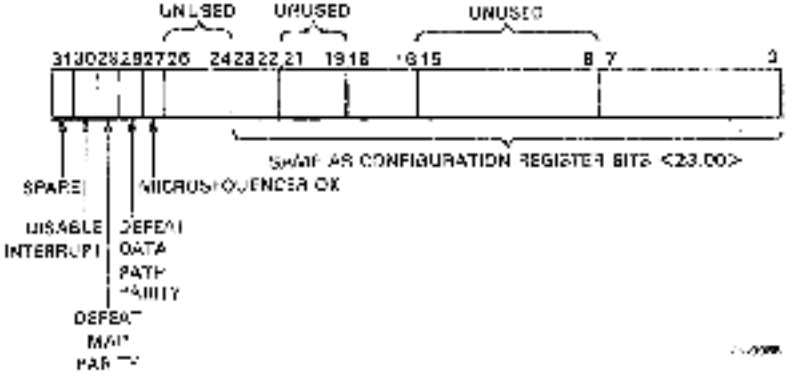


UBA REGISTERS

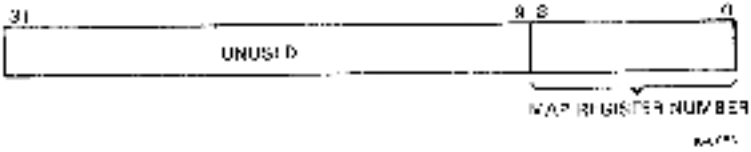
UBA STATUS REGISTER, BIT CONFIGURATION



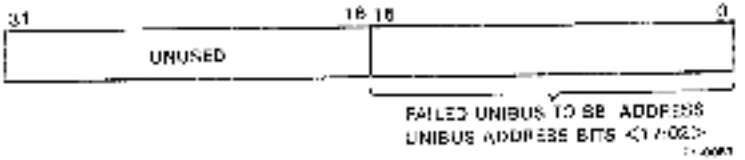
UBA DIAGNOSTIC CONTROL REGISTER, BIT CONFIGURATION



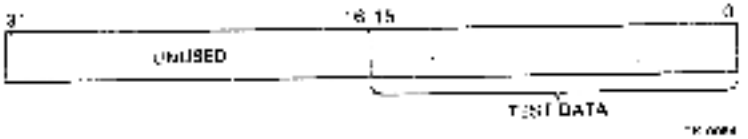
UBA FAILED MAP ENTRY REGISTER, BIT CONFIGURATION



UBA FAILED UNIBUS ADDRESS REGISTER, BIT CONFIGURATION

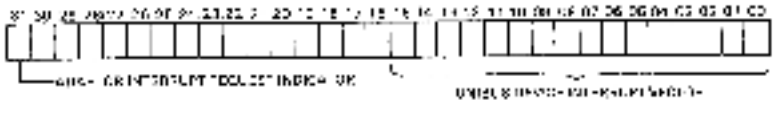


UBA BUFFER SELECTION VERIFICATION REGISTER, BIT CONFIGURATION

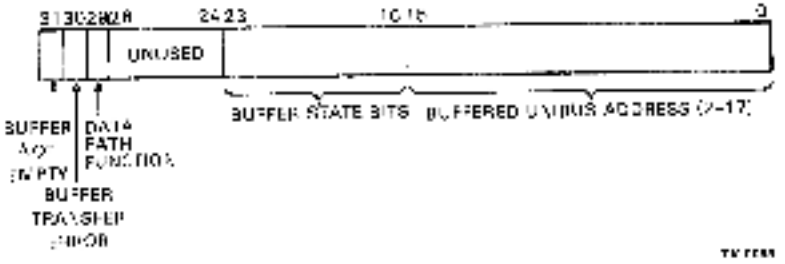


UBA REGISTERS

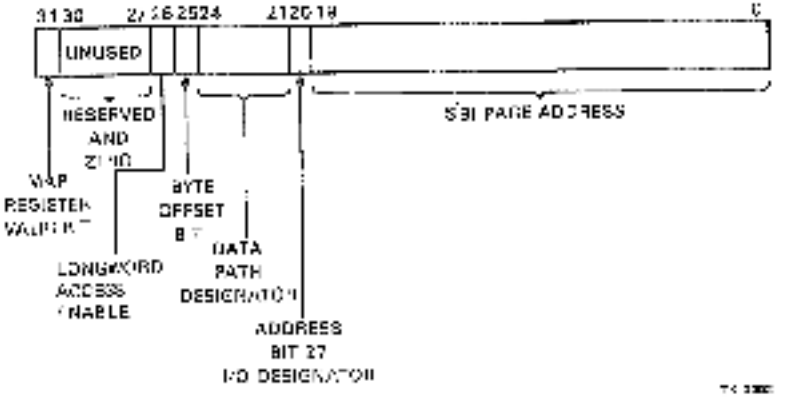
UBA BR RECEIVE VECTOR REGISTER, BIT CONFIGURATION



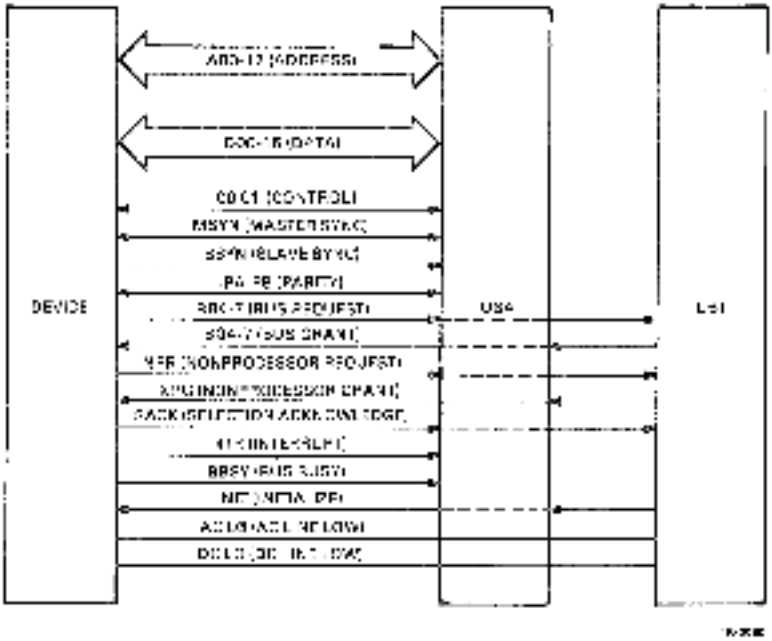
UBA DATA PATH REGISTER, BIT CONFIGURATION



UBA MAP REGISTER, BIT CONFIGURATION



UNIBUS CONFIGURATION



Signal Line	Description
Data Transfer Group	
Address Lines (A1 (17:01))	These lines are used by the master device to select the slave (usually a unique memory or device register address). SA (17:01) specifies a unique 16-bit word; BA00 specifies a byte within the word.
Data Lines (D (15:00))	These lines transfer information between master and slave.
Control (C1, C0)	These signals are driven by the master device to control the direction of the four possible data transfer operations specified below. Note that the master continues to always designated with respect to the master device.
C1, C0 1 1	Data In (DATI): a data word or byte transferred from the master from the slave.
0 1	Data In Pause (DATIP): similar to DATI except that it is always followed by a DATO(b) to the same location.
1 0	Data Out (DATO): a data word is transferred out of the master to the slave.
0 0	Data Out Byte (DATO(b)): identical to DATO except a byte is transferred instead of a full word.
Parity A, B (PA, PB)	These signals transfer Unibus parity information. PA is currently unused and not specified. PB, when true, indicates odd parity only.
Master Synchronization (MSYN)	MSYN is asserted by the master to indicate to the slave that valid address and control information (and data on a DATO or DATO(B)) is present on the bus.
Slave Synchronization (SSYN)	SSYN is asserted by the slave. On a DATO it indicates that the slave has latched the write data. On a DATI it indicates that the slave has asserted read data on the Unibus.
Interrupt (INTR)	This signal is asserted by an interrupting device after it becomes bus master to inform the CPU that an interrupt is to be performed, and that the interrupt vector is present on the D lines. INTR is negated upon receipt of the assertion of SSYN by the UBA at the end of the interaction. INTR may be asserted only by a device that otherwise has ownership under the authority of a BG signal.
Priority Arbitration Group	
Bus Request (BR7–BR4)	These signals are used by peripheral devices to request control of the bus for an interrupt operation.
Bus Grant (BG7–BG4)	These signals form the CPU and UBA response to a bus request. Only one of the four will be asserted at any time.

Signal Line	Description
Priority Arbitration Group (Cont.)	
Non-processor Request (NPR)	This is a bus request from a device for a transfer not requiring CPU intervention (i.e., DMA).
Non-processor Grant (NPG)	This is the grant in response to an NPR.
Selection Acknowledge (SACK)	SACK is asserted by a bus requesting controller following receipt of a grant. Its function is to inform the device which requested the bus that its request is being serviced.
Bus Busy (BBSY)	BBSY indicates that the data lines and the bus control lines are in use. It is asserted by the Unibus master.
Initialization Group	
Initialize (INIT)	This signal is asserted by the console terminal (UBI). After DC LO is asserted on the console, and it stays asserted for 10 milliseconds, INIT is asserted.
AC Line Low (AC LO)	This is an anticipatory signal that warns of an impending power failure. AC LO initiates the power fail (PF) sequence and may also be used to inform devices of temporary operations in preparation for power loss.
DC Line Low (DC LO)	This signal is used to inform each system power supply that it should shut down as long as all of its voltages are within the specified limits. If an out-of-voltage condition occurs, DC LO is asserted.

Pin	Standard Signal	Modified Signal
AA1	INIT L	NU
AA2	+5V	+5V
AB1	INTR L	INTR L
AB2	GROUND	TEST POINT
AC1	DOC L	DC0 L
AC2	GROUND	GROUND
AD1	DO2 L	DC2 L
AD2	DO1 L	DC1 L
AE1	DO4 L	DC4 L
AE2	DO3 L	DC3 L
AF1	DO6 L	DC6 L
AF2	DO5 L	DC5 L
AH1	DO8 L	DC8 L
AH2	DO7 L	DC7 L
AJ1	DI0 L	DC0 L
AJ2	DO9 L	DO9 L
AK1	DI2 L	DC2 L
AK2	DI1 L	DC1 L
AL1	DI4 L	DC4 L
AL2	DI3	DC3 L
AM1	DI6 L	DC6 L
AM2	DI5 L	DC5 L
AN1	GROUND	PA*
AN2	PB L	PB L

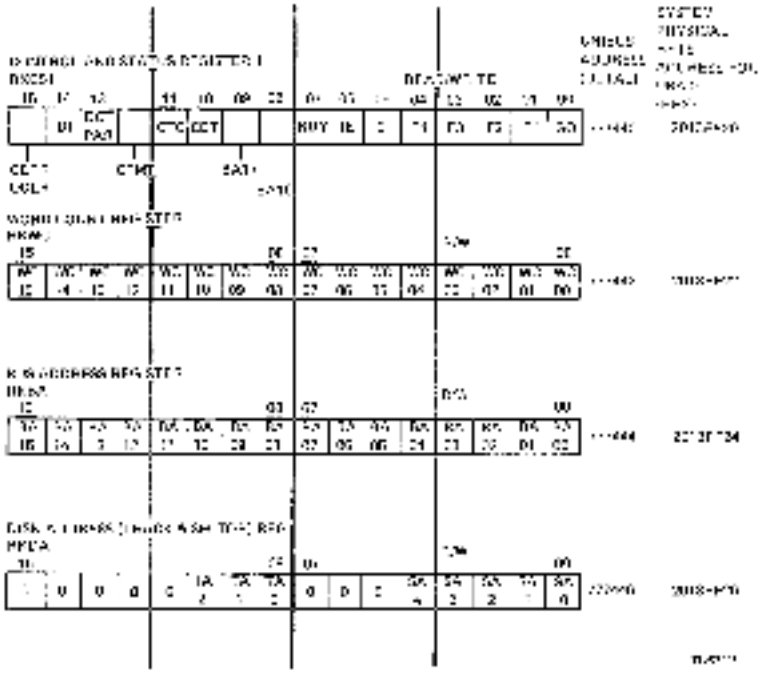
Pin	Standard Signal	Modified Signal
AP1	GROUND	PC*
AP2	BUSY L	BUSY L
AR1	GROUND	BAT BACKUP +3.3V
AR2	SACK L	SACK L
AS1	GROUND	BAT BACKUP +3.3V
AS2	NPR L	NPR L
AT1	GROUND	GROUND
AT2	BR7 L	BR7 L
AU1	NPG I	+20V
AU2	BR6 L	BR6 L
AV1	BG7 H	+20V
AV2	GROUND	+20V
BA1	BG6 H	SPARE
BA2	+5V	+5V
BB1	BG5 H	SPARE
BB2	GROUND	TEST POINT
BC1	BR5 L	BR5 L
BC2	GROUND	GROUND
BD1	GROUND	BAT BACKUP +5V
BD2	BR4 L	BR4 L
BE1	GROUND	SYNC SYNC*
BE2	BG7 H	PARITY*
BF1	ACLO L	ACLO L
BF2	DCLO L	DCLO L

Pin	Standard Signal	Modified Signal
BH1	A01 L	A01 L
BH2	A00 L	A00 L
BJ1	A03 L	A03 L
BJ2	A02 L	A02 L
BK1	A05 L	A05 L
BK2	A04 L	A04 L
BL1	A07 L	A07 L
BL2	A06 L	A06 L
BM1	A09 L	A09 L
BM2	A08 L	A08 L
BN1	A11 L	A11 L
BN2	A10 L	A10 L
BP1	A13 L	A13 L
BP2	A12 L	A12 L
BR1	A15 L	A15 L
BR2	A14 L	A14 L
BS1	A17 L	A17 L
BS2	A16 L	A16 L
BT1	GROUND	GROUND
BT2	C1 L	C1 L
BU1	SSYN L	SSYN L
BU2	C0 L	C0 L
BV1	MSYN L	MSYN L
BV2	GROUND	+5V

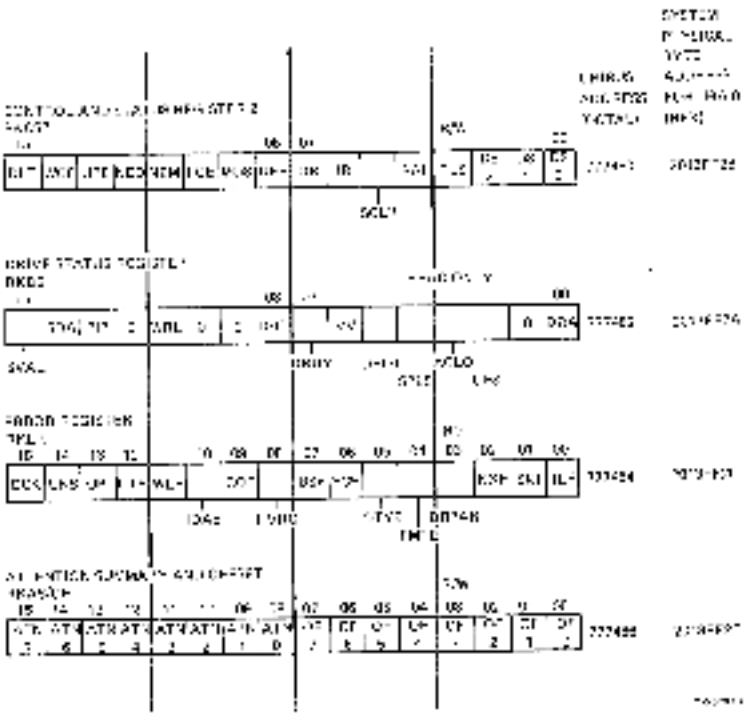
* Available by special request.

UBA #	DEVICE	UNIBUS ADDRESS (OCTAL)	SBI ADDRESS (HEX)	11/780 PHYSICAL ADDRESS (HEX)	VECTORS (HEX)
0	CR11/CR01	777160	08004FF0	20140F70	98
"	CR11/CR01	777162	08004FF0	20140F72	
0	CR11/CR02	777164	08004FF3	20140F74	
0	CP11/LPOT	777514	08004FED	20140F8C	80
"	CP11/LPDR	777516	08004FED	20140F8F	
"	RK611C01	777440	08004FF0	20140F70	(73G)
"	RK611 MC	777442	08004FC0	20140F22	
"	RK611 BA	777444	08004FC0	20140F24	
"	RK611 DA	777446	08004FC0	20140F26	
0	RK611 CS2	777450	08004FF0	20140F28	
0	RK611 DB	777452	08004FF0	20140F2A	
0	RK611 DBR	777454	08004FF0	20140F2C	
0	RK611 ATA	777456	08004FF0	20140F2E	
0	RK611 CYL	777460	08004FF0	20140F30	
"	UNUSED	777462	08004FF0	20140F32	
"	RK611 DB	777464	08004FF0	20140F34	
0	RK611 MR1	777466	08004FF0	20140F36	
0	RKECR1	777470	08004FF0	20140F38	
0	RKECPT	777472	08004FF0	20140F3A	
0	RK611 MR2	777474	08004FF0	20140F3C	
"	RK611 MR3	777476	08004FF0	20140F3E	
0	DZ11	Floating addresses and vectors			
0	DMC11	according to DEC-11			
0	DR11-B	convention			

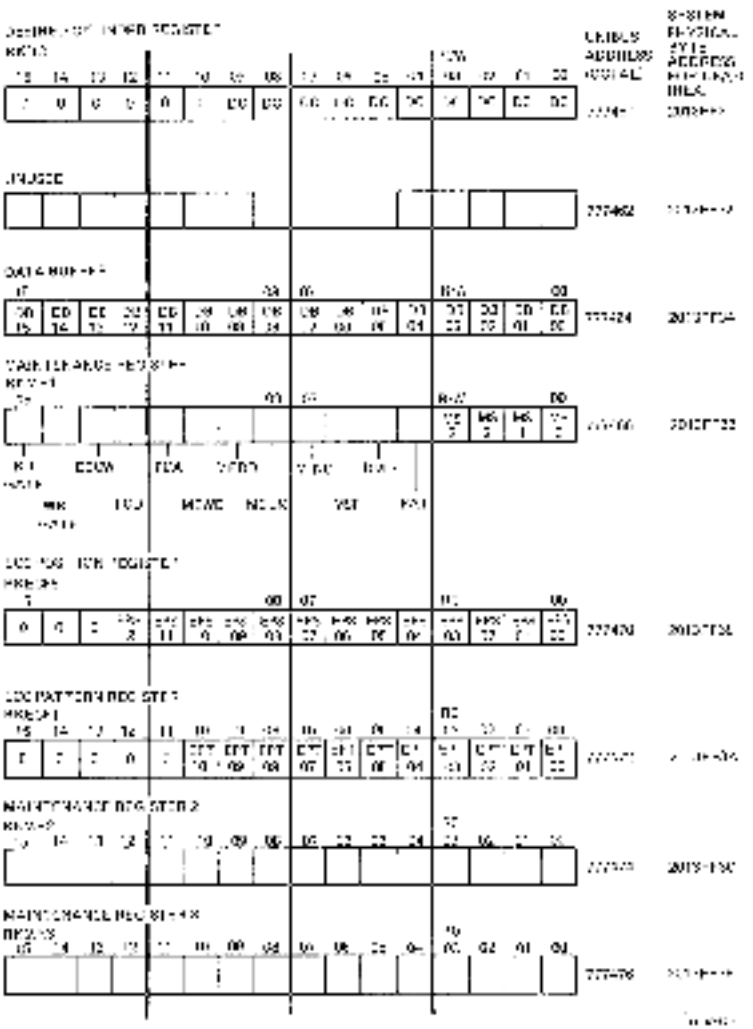
RK611 REGISTER CONTENTS



RK611 REGISTER CONTENTS

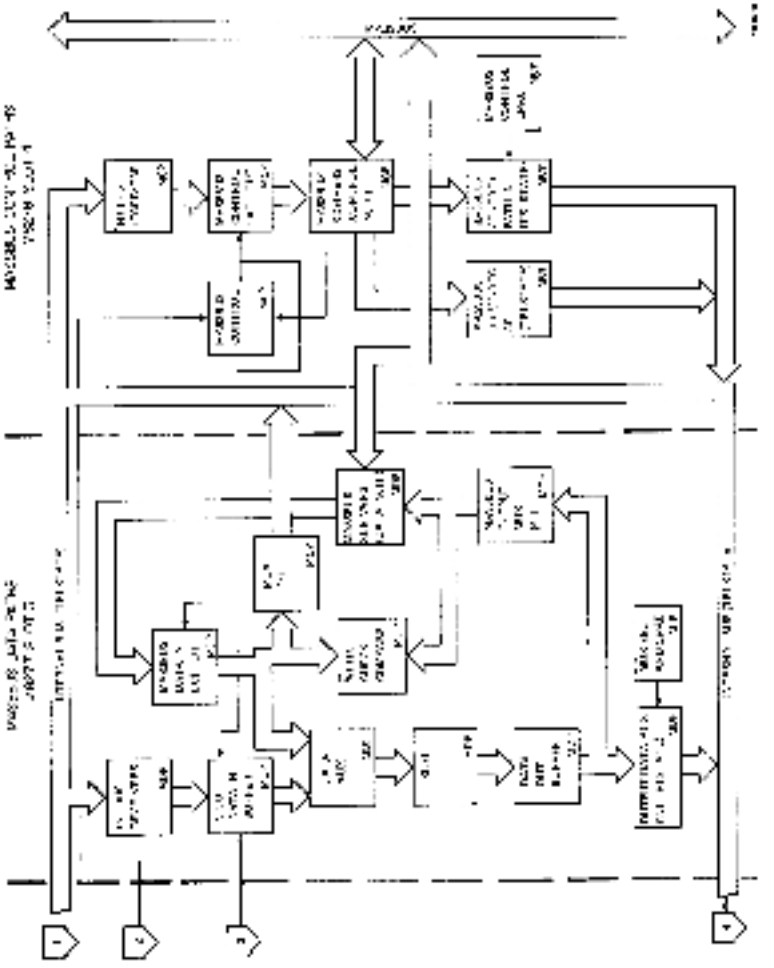


RK611 REGISTER CONTENTS



MBA (KH78K) BLOCK DIAGRAM, PART 1

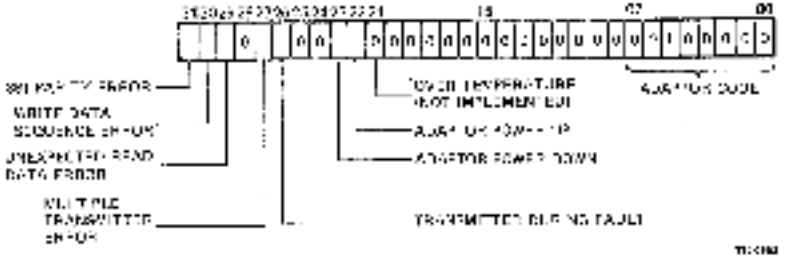
MGA (RH780) BLOCK DIAGRAM, PART 2



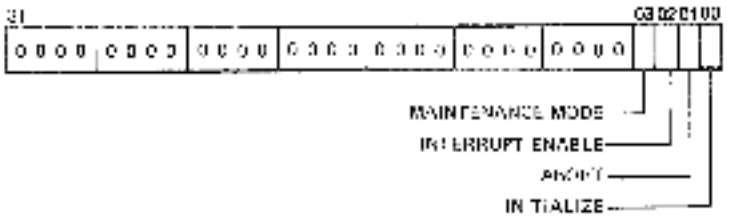
OFFSET FROM LINK SBI ADDRESS	REGISTER	OFFSET FROM BASE PHYSICAL ADDRESS	
00	Configuration Register (CSR)	00	R/W
01	Control Register (CR)	04	R/W
02	Status Register (SR)	08	R/W
04	Virtual Address Register (VAR)	0C	R/W
05	Byte Counter Register (BCR)	10	R/W
06	Diagnostic Register (DR)	14	R/W
07	Selected MAP Register (SMR)	18	Read only
08	Command/Address Register (CAR)	1A	Read only

MBA REGISTERS

MBA CONFIGURATION/STATUS REGISTER, BIT CONFIGURATION



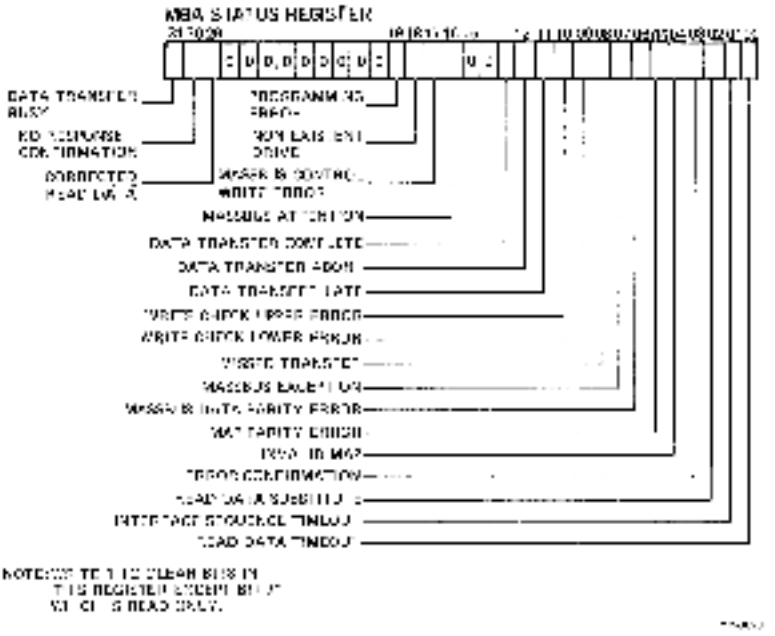
MBA CONTROL REGISTER, BIT CONFIGURATION



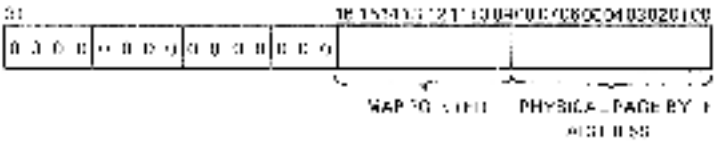
NOTE: ALL BITS ARE READ/WRITE EXCEPT BIT 02 WHICH IS WRITE ONLY AND ALWAYS READS AS 0

MBA REGISTERS

MBA STATUS REGISTER, BIT CONFIGURATION

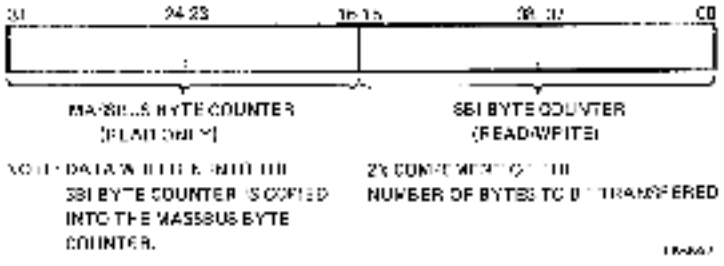


MBA VIRTUAL ADDRESS REGISTER, BIT CONFIGURATION

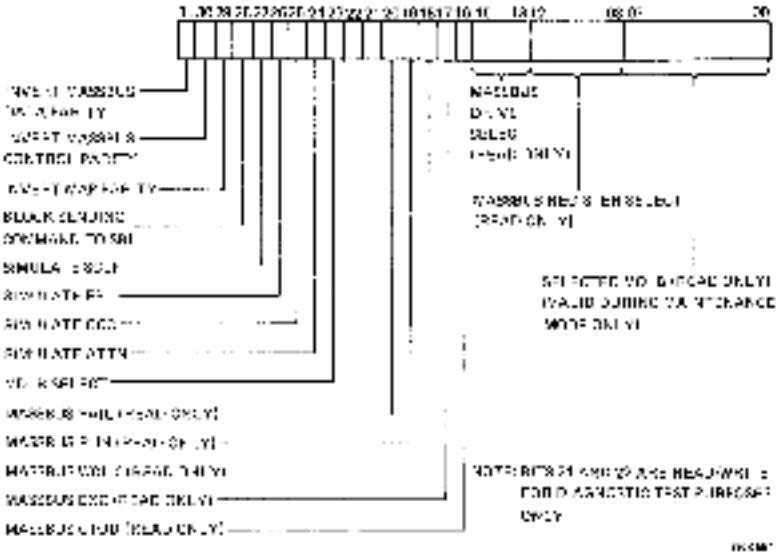


MBA REGISTERS

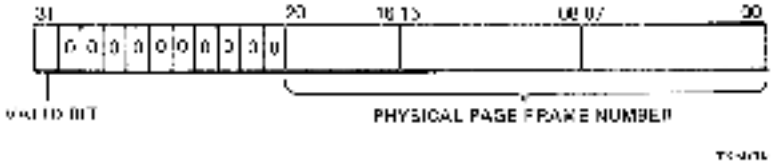
MBA BYTE COUNTER REGISTER, BIT CONFIGURATION



MBA DIAGNOSTIC REGISTER, BIT CONFIGURATION



MBA MAP REGISTER, BIT CONFIGURATION



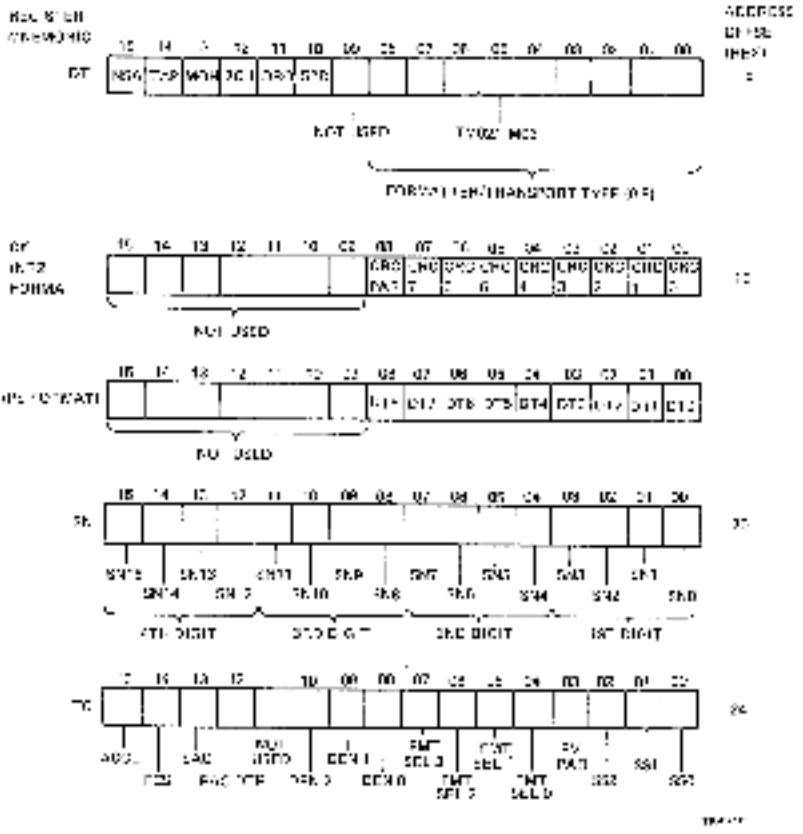
MASSBUS DISK DRIVE REGISTER ADDRESS CALCULATION CHART

1st SDA Base Address 20012400
2nd MBA Base Address 20012400

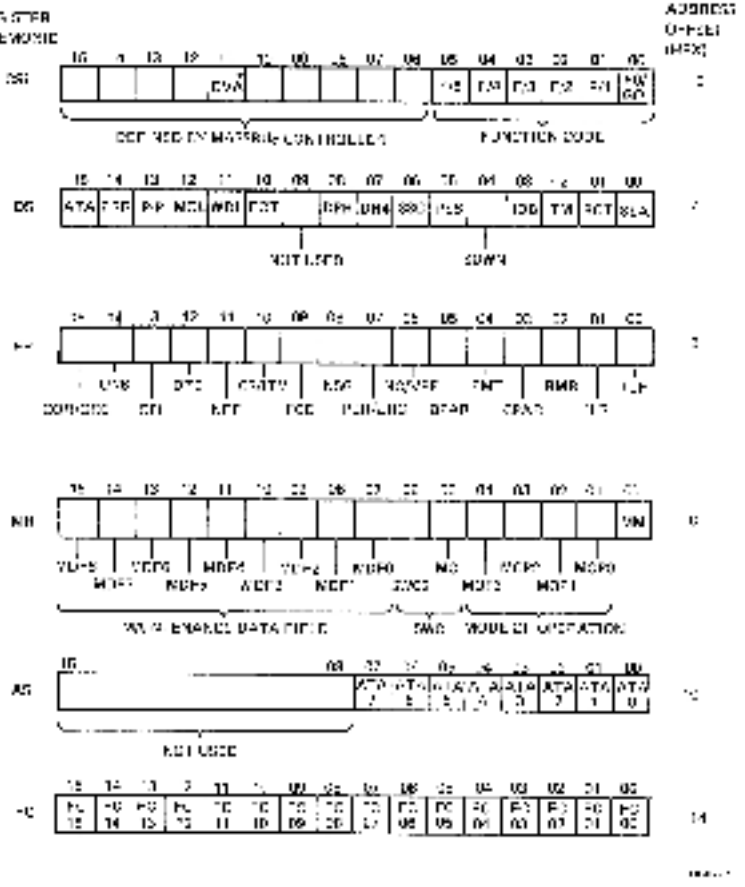
REGISTER NUMBER		DRIVE	TYPE		DRIVE NUMBER							
HEX	OCTAL	RP (DTSE)	RM (DTSE)	TU (TMSE)	0	1	2	3	4	5	6	7
0	0	CS1	RMCS1	CS1	80	80	100	180	200	280	300	380
1	1	DS	RMDS	DS	84	84	104	184	204	284	304	384
2	2	ER1	RMER1	ER	88	88	108	188	208	288	308	388
3	3	MR	RMMR1	MR	8C	8C	10C	18C	20C	28C	30C	38C
4	4	AS	RMAS	AS	90	90	110	190	210	290	310	390
5	5	DA	RMDA	FC	94	94	114	194	214	294	314	394
6	6	DT	RMDT	DT	98	98	118	198	218	298	318	398
7	7	LA	RMLA	CK	9C	9C	11C	19C	21C	29C	31C	39C
8	10	SN	RMSN	SN	A0	A0	120	1A0	220	2A0	320	3A0
9	11	OFF	RMOF	TC	A4	A4	124	1A4	224	2A4	324	3A4
A	12	DCA	RMDC		A8	A8	128	1A8	228	2A8	328	3A8
B	13	CCA	RMMR		AC	AC	12C	1AC	22C	2AC	32C	3AC
C	14	ER2	RMMR2		B0	B0	130	1B0	230	2B0	330	3B0
D	15	ER3	RMER2		B4	B4	134	1B4	234	2B4	334	3B4
E	16	EC2	RMEC1		B8	B8	138	1B8	238	2B8	338	3B8
F	17	MR2	RMEC2		BC	BC	13C	1BC	23C	2BC	33C	3BC
-	-	-	-		-	-	-	-	-	-	-	-
-	-	-	-		-	-	-	-	-	-	-	-
1F	37				FC	FC	17C	1FC	27C	2FC	37C	3FC

RP05/RP06 REGISTER CONTENTS

TM903 REGISTER CONTENTS



TM03 REGISTER CONTENTS



MASSBUS SIGNAL CABLE PIN ASSIGNMENTS

Massbus Signal Cable Designations

Cable	Pair		Polarity	Designation
Massbus Cable A	A	1	-	MASS DA0
	B	2	+	
	C	3		MASS DA1
	D	4	-	
	E	5	-	MASS DA2
	F	6	+	
	H	7	+	MASS DA3
	J	8	-	
	K	9	-	MASS DA4
	L	10	+	
	M	11	+	MASS DA5
	N	12	-	
	P	13	-	MASS DA6
	R	14	+	
	S	15	+	MASS DA7
	T	16	-	
	U	17	-	MASS DA8
	V	18	+	
	W	19	+	MASS DA9
	X	20	-	
	Y	21	-	MASS DA10
	Z	22	+	
	AA	23	+	MASS DA11
	BB	24	-	
	CC	25		MASS SPARE
	DD	26	-	
	EE	27	-	MASS DA12
	FF	28	+	
	HH	29	+	MASS DA13
	JJ	30	-	
	KK	31	-	MASS DA14
	LL	32	+	
	MM	33	+	MASS DA15
	NN	34	-	
	PP	35	-	MASS DA16
	RR	36	+	
	SS	37		MASS DA17
	TT	38	-	
	UU	39		MASS SPARE
	VV	40		GND

Massbus Signal Cable Designations

Cable	Pair		Polarity	Designation
Massbus Cable B	A	1	-	MASS DB0
	B	2	+	
	C	3	+	MASS DB1
	D	4	-	
	E	5		MASS DB2
	F	6	+	
	H	7	+	MASS DB3
	J	8	-	
	K	9		MASS DB4
	L	10	+	
	M	11	+	MASS DB5
	N	12	-	
	P	13		MASS DB6
	R	14	+	
	S	15	+	MASS DB7
	T	16	-	
	U	17		MASS DB8
	V	18	-	
	W	19	+	MASS DB9
	X	20	-	
	Y	21		MASS DB10
	Z	22	-	
	AA	23	-	MASS DB11
	BB	24		
	CC	25		MASS SPARE
	DD	26	-	
	EE	27		MASS DB12
	FF	28		
	HH	29	+	MASS DB13
	JJ	30	-	
	KK	31		MASS DB14
	LL	32	+	
	MM	33	-	MASS DB15
	NN	34	+	
	PP	35		MASS DB16
	RR	36		
	SS	37	+	MASS DB17
	TT	38		
	UU	39		SPARE
	VV	40		GND

Note: [illegible]

Note: [illegible]

MASSBUS SIGNAL CABLE PIN ASSIGNMENTS

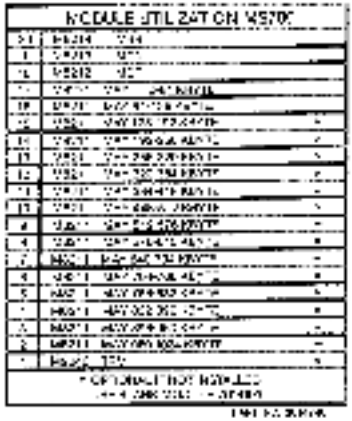
Massbus Signal Cable Designations

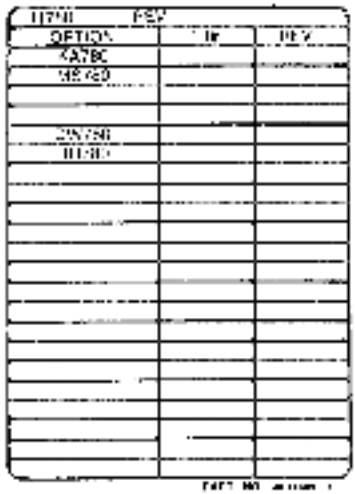
Cable	Pair*		Polarity	Designation
Massbus Cable C	A	1		MASS 01 L
	B	2	+	
	C	3	–	MASS 01 H
	D	4	–	
	E	5		MASS 01–
	F	6	+	
	H	7	–	MASS 01
	J	8	–	
	K	9		MASS 016
	L	10	–	
	M	11	+	MASS 017
	N	12	–	
	P	13		MASS 00A
	R	14	–	
	S	15	–	MASS C 2
	T	16		
	U	17	–	MASS C 4
	V	18		
	W	19	–	MASS C14
	X	20	+	
	Y	21		MASS C15
	Z	22	+	
	AA	23	+	MASS P24
	BB	24	–	
	CC	25		MASS C2L
	DD	26	+	
	EE	27	+	MASS C2
	FF	28		
	HH	29		MASS E2A
	JJ	30		
	KK	31	–	MASS 021
	LL	32	+	
	MM	33		MASS 022
	NN	34	+	
	PP	35	–	MASS 023
	RR	36	–	
	SS	37	+	MASS 012
	TT	38		
	UU	39	H	MASS Ext
	VV	40		GND

*Wires in pair are adjacent numbers

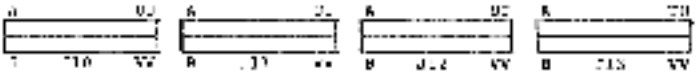
SECTION 4
CONFIGURATION JUMPERS

MODULE UTILIZATION CHART





KA780 TR, SYSID REGISTER JUMPERS



TR Arbitration Level

SIGNAL NAME	TR SEL 8	TR SEL 4	TR SEL 2	TR SEL 1
SID	J10 C	J11 B	J10 A	J10 9
1	--	--		
2	--	--	--	1
3	--	--	1	
4	--	--	1	
5	--	1	--	--
6	--	1	--	1
7	--	1	1	--
8	--	1	1	1
9	1	--	--	--
10	1	--	--	1
11	1	--	1	--
12	1	--	1	1
13	1	1	--	--
14	1	1	--	1
15	1	1	1	--
16	1	1	1	1

WIRE WRAP PIN3 to

E0321
E1201
E0281
E1202
E0752
E0211
E0331
E0201
E0202
E0203
E0204
E0205
E0206
E0207
E0208
E0209
E0210
E0211
E0212
E0213
E0214
E0215

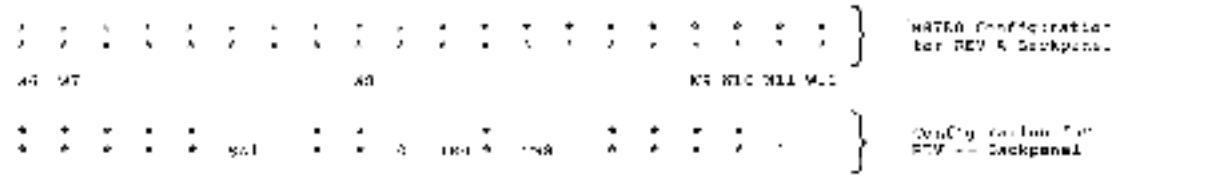
System ID Register
Remove Jumper to Assert bit

Bit	Jumper		Signal
0	J11	TT	
1	J11	SS	
2	J11	RR	
3	J11	QQ	
4	J10	LL	SYSTYP
5	J10	KK	SBIFAC
6	J10	FF	A.R.Fail
7	J10	DD	
8	J10	CC	
9	J10	Z	
10	J10	Y	
11	J10	X	
12	J10	V	
13	J10	U	CPU
14	J10	T	START
15	J10	L	Lock
16	J10	J	
17	J11	H	
18	J11	G	CPU
19	J11	F	CLUSTER
20	J11	EE	ECO
21	J11	II	LEVEL
22	J11	BB	
23	J11	AA	
24	J11	**	
25	J12	JJ	SYSTEM
26	J12	KK	SYSA
27	J12	LL	01-11/51
28	J12	MM	00=0
29	J12	N	01=1
30	J12	P	
31	J12	R	

	WCS SLOT 20 J11					OPTIONAL WCS SLOT 18 J11				
	VV	TT	RR	NN	LL	FF	DD	BB	Z	X
0-1K	I	I	I	I	--	I	I	I	I	--
1-2K	I	I	I	--	I	I	I	I	--	I
2-3K	I	I	--	I	I	I	I	--	I	I
3-4K	I	--	I	I	I	I	--	I	I	I
4-5K	--	I	I	I	--	--	I	I	I	--
5-6K	--	I	I	--	I	--	I	I	--	I
6-7K	--	I	--	I	I	--	I	I	I	I
7-8K	--	--	I	I	I	--	I	I	I	I

PCS
SLOT 22
J12

	L	J	K	D	B
0-4K	--	--	--	--	--



VAX Arbitration Level

SIGNAL NAME	IR SEL 0	IP SEL 1	IN SEL 2	IR SEL 3
DRP	MJ	MID	MJL	MJ2
0				1/2
1				
2	--	--	--	
3	--	--	--	
4	--	--		
5		1		
6	—	1	--	
7		1		
8	—	1	1	
9		--	--	--
10		--	--	
11		--		--
12			1	
13		1	--	--
14				
15		1	1	--

WIRE WRAP PIN ASSIGN
P20 C1
P20 A1
P20 C2
P20 A2
P20 C3
P20 A3
P20 C4
P20 A4
P20 C5
P20 A5
P20 C6
P20 A6
P20 C7
P20 A7
P20 C8
P20 A8

1 = Standard for Memory 1
2 = Standard for Memory 2

Top Side BDR Decode

MB=1 — Bus Arbitration Configuration for Read to BDR Address Space

MB=0 — Read to BDR[1] Address Space receive Normal confirmation

NOTE: When installing 2 MS780A and an MS7800 on the same VAX-11/780 system, the MS781C must be the first memory (i.e., lowest TR).

Stacking Access

	MJ	MJ2
0	--	--
4 Mega Byte	--	1
8 Mega Byte	1	
12 Mega Byte	1	1

MJ, MJ Space

[2]

ARRAY	M8211 SIZE	ADDRESS RANGE	M8210 SIZE	ADDRESS RANGE
1	64 K	0- 3FFF	256 K	0- 3FFFF
2	128 K	10000-1FFFF	512 K	40000 7FFFF
3	192 K	20000-2FFFF	768 K	80000- BFFFF
4	256 K	30000-3FFFF	1024 K	C0000- FFFFF
5	320 K	40000-4FFFF	1280 K	100000-13FFFF
6	384 K	50000-5FFFF	1536 K	140000-17FFFF
7	448 K	60000-6FFFF	1792 K	180000-1BFFFF
8	512 K	70000-7FFFF	2048 K	1C0000-1FFFFF
9	576 K	80000-8FFFF	2304 K	200000-23FFFF
10	640 K	90000-9FFFF	2560 K	240000-27FFFF
11	704 K	A0000-AFFFF	2816 K	280000-2BFFFF
12	768 K	B0000-BFFFF	3072 K	2C0000-2FFFFF
13	832 K	C0000-CFFFF	3328 K	300000-33FFFF
14	896 K	D0000-DFFFF	3584 K	340000-37FFFF
15	960 K	E0000-EFFFF	3840 K	380000-3BFFFF
16	1024 K	F0000-FFFFF	4096 K	3C0000-3FFFFF

Memory Starting Address Jumpers

Boundary	Address
0	000000
4 MEG	400000
8 MEG	800000
12 MEG	C00000

(Memory Array Boards (MB814, MB817))

SYN BITS 7-0	SYN	SYN BITS 3-0 (BIT IN ERROR)						
	BYTE	111	110	101	100	011	010	001	000
	LOWER								
00*11	000	7	6	5	4	3			2
01*01	001	15	14	13	12	11	10	9	8
01*10	002	23	22	21	20	19	18	17	16
01*11	003	*	30	29	28	27	26	25	24
	UPPER								
10*01	004	7	6	5	4	3	2	1	0
10*10	005	15	14	13	12	11	10	9	8
10*11	006	23	22	21	20	19	18	17	16
11*00	007	31	30	29	28	27	26	25	24
00100	CHECK				C2		C1	C0	
00001	CHECK								C1
00010	CHECK								C2
00100	CHECK								C3
01000	CHECK								C6
10000	CHECK								C7

* DON'T CARE ON THE VALUE OF THIS BIT (USED ONLY TO SKIP AN EVEN NUMBER OF CHECK BITS PRESENT AT ONE TIME).

RMBOR LOGOUT

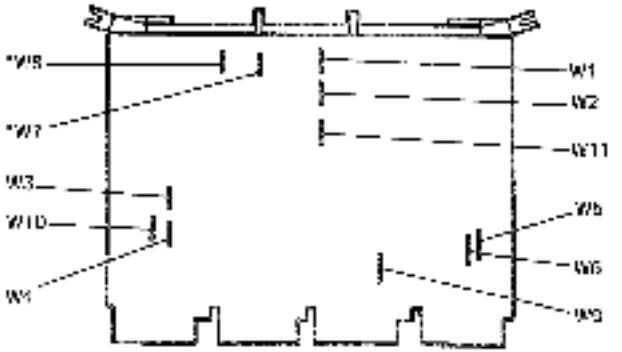
DATA: UPPER BITS IN ERROR
LOW BITS IN ERROR
SYNDROME BITS READ
FAILURE ADDRESS

DESCRIPTION	BIT POSITION 4K CHIP	16K CHIP
BYTE POSITION	2-0	2-0
CHIP ADDRESS	14-3	16 >
BANK SELECT	15	17

DW780 (UBA) BACKPANEL JUMPER CONFIGURATION

RH780 (MBA) BACKPANEL JUMPER CONFIGURATION

JUMPER	IN/OUT	RESULT
W1	– –	RESIDENT MEMORY AT A BANK 0
W2	– I	RESIDENT MEMORY AT A BANK 0
W3		LTC INTERRUPT ENABLED
W4	– –	MEMORY REFRESH ENABLED
W5	– –	POWER UP AT 173000
W6	–	POWER UP AT 173000
W7 & W8		PREDONFIGURED*
W9	– –	ENABLE REPLY FROM RESIDENT MEMORY
W10	– I	DISABLE REPLY FROM RESIDENT MEMORY DURING REFRESH
W11	– I	ENABLE ON BOARD MEMORY SELECT

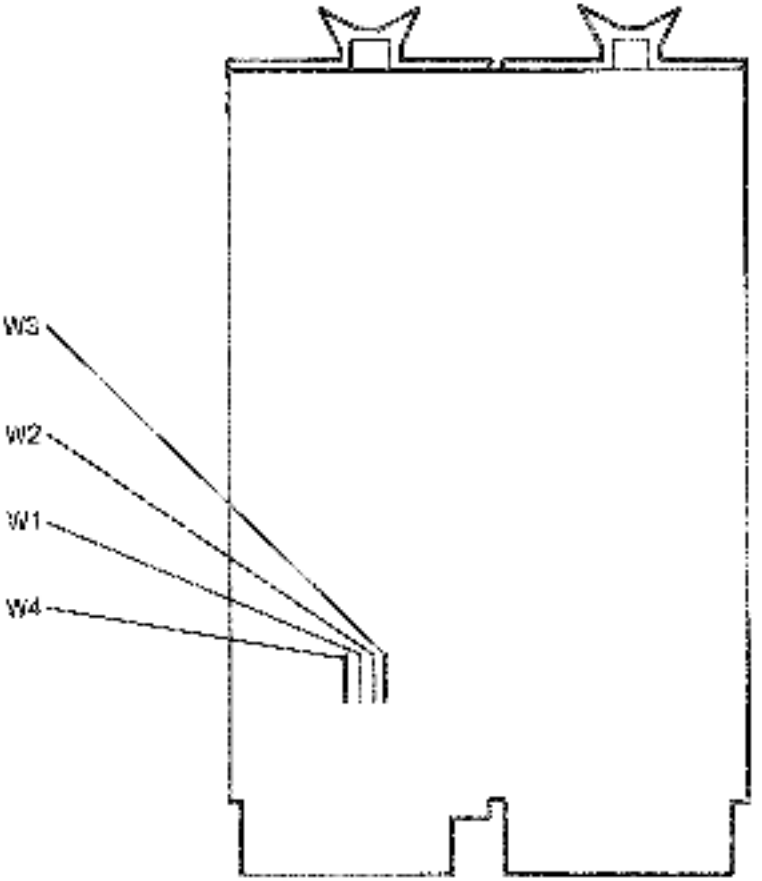


M7264 ETCH REV. E (AND LATER)

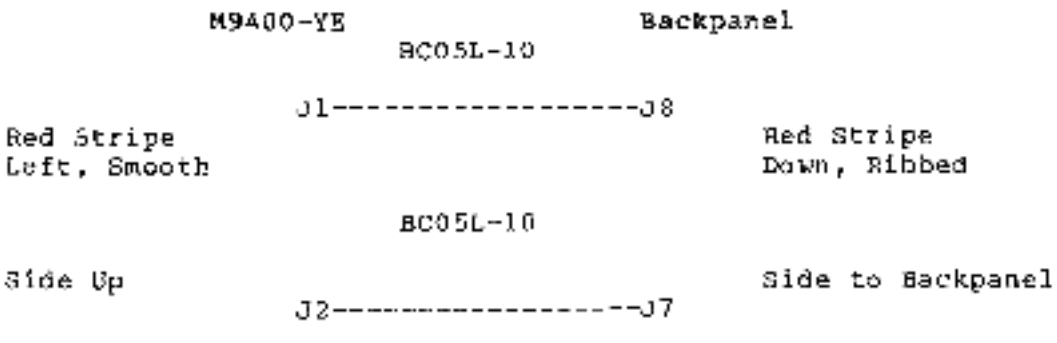
*FACTORY CONFIGURED DO NOT CHANGE (W7 & W8)

MSV-11S MODULE JUMPER CONFIGURATION

JUMPER	IN/OUT	RESULT
W1	I	MEMORY BANK SELECT 1
W2	I	(20000-37776)
W3	–	
W4	–	ENABLES BRPLY DURING REFRESH



TK-0702



Configuration of RXV-11 (M7945)

Should be preconfigured for:

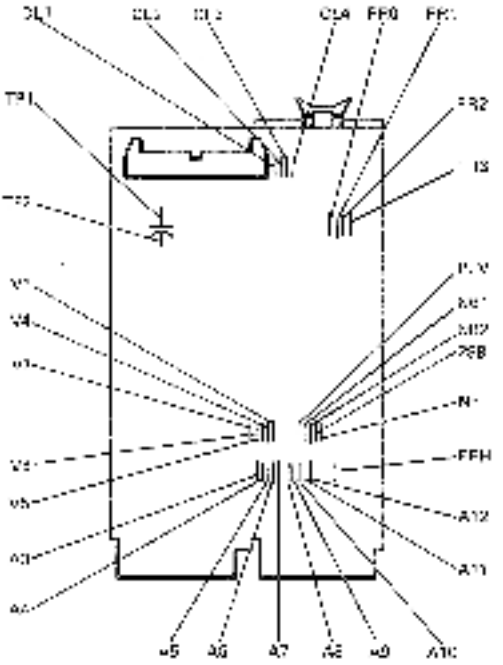
Address: 177170-177172
Vector: 264
Ribbon cable should be installed with
red stripe toward center of module.

DLV11 JUMPER CONFIGURATION

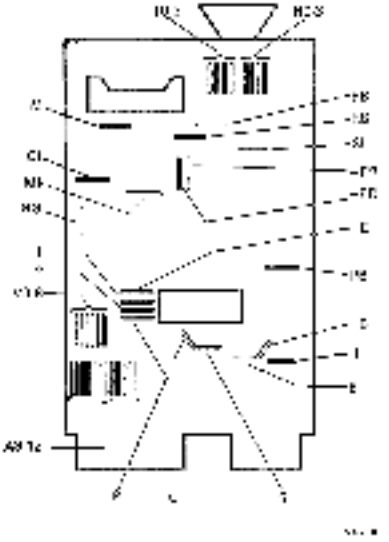
JUMPER	IN/OUT	RESULT
	—	NO PARITY
2SS	I	1 STOP BIT
NB2		8 DATA BITS
NB1	—	7 DATA BITS
ODV	X	DON'T CARE PARITY EVEN/ODD
PEN	X	ADDITIONAL PARITY/FRAMING ERROR
SIA		NO SIA OPERATION
FR3	—	SELECTS 9600 BAUD
FR2	—	SELECTS 4800 BAUD
FR1	I	SELECTS 300 BAUD
DMAC/C	I	DMA ACTIVE (BUS GRANT)

VECTOR JUMPERS: 300 TO 304
V7 I, V6 I, V5 I, V3 I

ADDRESS JUMPERS SELECTED: 176500 - 176506
A12 —, A11 —, A10 —, A9 —, A8 —,
A7 I, A6 I, A5 I, A4 I, A3

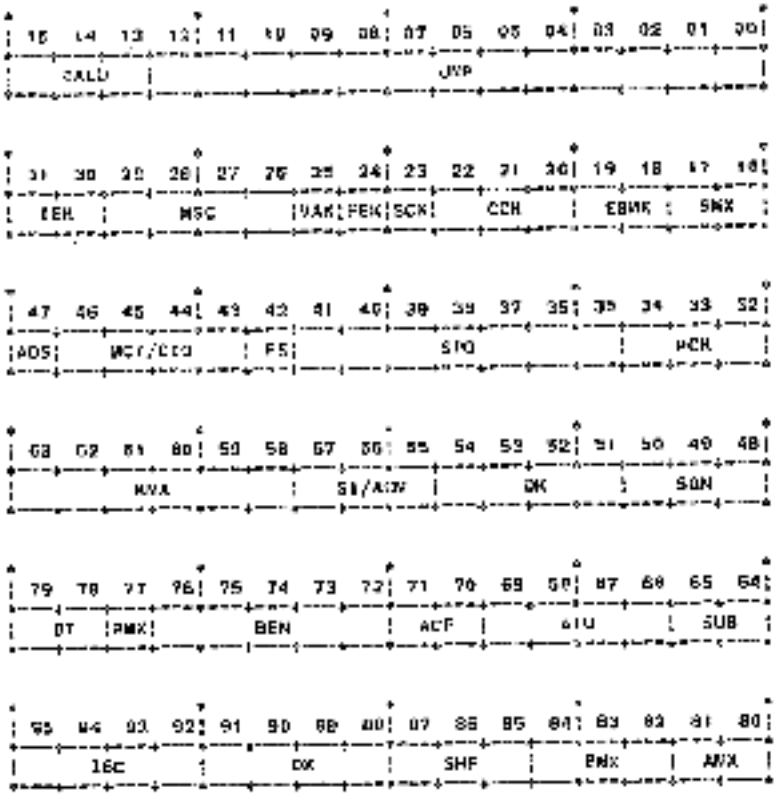


DLV11-E JUMPER CONFIGURATION



SECTION 5
CONTROL STORE

CONTROL STORE FIELD MAP



MICROCODE ROUTINES WHICH SUPPORT CONSOLE SOFTWARE, STARTING ADDRESSES

```
'CONSOLE MICRO-CODE'


;MICRO-CODE ROUTINES TO SUPPORT CONSOLE SOFTWARE.
;STARTING ADDRESS GIVEN IN MACRO, AND IN TD[T1],TD[T3]
;ON RETURN STATUS IN TD[0.5W].
;ON MOST CALLS INFORMATION IN TD[T3].
;IF IN JOB, USERPAGE R-S IS PRESERVED.
;AD 8--341 IS USED TO SAVE INTERNAL REGISTERS.

;INFORMATION AND PARAMETERS NEEDED FROM THE CONSOLE.
;ARG 10-0&1 IN TD[AXCD] AND TD[T1,,T2].
;RESULTING DATA IS LOADED IN TD['ACD] AND TD[T3].
;AND STATUS INFORMATION IS LOADED IN TD[0.5W].
```

;ROUTINE	START-ADDRESS	PARAMETERS (& RESULTS SUPPLIED BY CONSOLE)
;EXAMINE MEMORY	130	TD[T1]=8-TS/KOMD/,IND=PARAMETER * TD[PCD]=VIRTUAL ADDRESS * TD[T2D]=MEMORY DATA TD[T3]=MEMORY BLANKS TD[0.5W]=STATUS CODE
;DEPOSIT MEMORY	131	TD[T1]=K/B/N/A/,IND=PARAMETER * TD[PCD]=VIRTUAL ADDRESS * TD[T2]=MEMORY DATA * TD[TA3D]=PHYSICAL ADDRESS TD[0.5W]=STATUS CODE
;PARALOAD.REG.	132	TD[PACD]=REGISTER NUMBER * TD['ACD]=REGISTER DATA
;CFG.GEN.REG.	133	TD[PACD]=REGISTER NUMBER * TD[T2]=REGISTER DATA *
;EXAM.PREV.REG.	134	TD[PACD]=REGISTER NUMBER * TD[T2]=REGISTER DATA
;DEP.PREV.REG.	135	TD[PACD]=REGISTER NUMBER * TD[T2]=REGISTER DATA *
;CONTINUE	136	
;LOAD-PROGRAM	137	TD[PACD]=LOAD-ADDRESS *
;SET-UP/AM	138	

MICRO II(230) STAR Microcode: FDS 01, FDJ 04, VDS170- PAB 14, IS
Branch Enable Functions

Brx-0 bypass Functions* 21-January-87

(BEN) Name	UPC(02)	uPC(01)	UPC(02)
0	ADF	0	0	0
1	2	0	0	PLL 2
2	4M	Loop	MSI<0>	Loop
3	CD*	0	=LU CDI	0
4				
5				
6	ACCa'enable*	ACC U32	=CC U31	=CC U30
7				
8 (AA)	DATA type 0 Move A 1 = A + B 2= A - B ** 0 + 0 3= Acer Src	Sreg + Modify	=SRC - =STC	=SPC + 0 1 0
9 (+1)	SYC CDI	Reco + Modify	= 0 = 1 0	0 0 = = + UPDT
A (AA)	IRD-1	0	IR=0*	IRY 0
B (+1)	PC Moves	0	NP => IR Y- = SY	0 1 M Acc. PC
C	RET	Mode ctl. =ST.A.		
D	IB TEST 0= IB Bit 1= IB Bit 2= IB Bit 3= IB Bit 4	0	B => 0-0-0	.4 =Mode + Data Ctrl.D
E	M..	IC.ne.3	D=IO	D=EC*
F	SIO=S	D=S Y	0 1 = 1 0	D=N*
G	(1)(HOLD)	NO Use	.Data-Fail Interrupt	Interrupt Request
H	Decimal	0	D=1 EC* IC=N	D=1 EC* HA = 1

MICROCODE BRANCH ENABLE FUNCTIONS

BRA	NAME	UPC<03>	UPC<02>	UPC<01>	UPC<00>
10	uProg Vector	LVEC<03>	LVEC<02>	LVEC<01>	LVEC<00>
11	LOOP M Counter	MCNT<04>	Decimal Error	MP Chip Error	Sequence Error
12	ALU OE	ALU N	ALU Z	SC.ne.0	Stack Src
13					
14	SC = 0= Zero = 1= Not Zero Zero 2= Not Zero	0	SC<0:0> .ne.0	SC.gt.0	SC<0:0> .ne.0
15	ALU = Intermediate Cycle	M = 0 for Y	ALU<01> .OR.S	ALU<01>	ALU<00>
16	STATE<7:3>	STATE<7>	STATE<6>	STATE<5>	STATE<4>
17	STATE<2:0>	STATE<2>	STATE<2>	STATE<1>	STATE<0>
18	I Bytes	Byte 1 .ne.0	Byte 0 .ne.0	Byte 1 .ne.0	Byte 0 .ne.0
19	IR=H	D<3>	D<2>	D<1>	D<0>
1A	FSL CC	FSL<3>	FSL<2>	FSL<1>	FSL<0>
1B	ALU CC	ALU V	ALU Z	IR<0>	ALU C

BRA	Name	UPC<13>	UPC<12>	UPC<11>	UPC<10>
1C	MAP VALUE	MAP<3>	MAP<2>	Console Mode	FSL<13> FSL<12>	Kernel Mode
D	Translation Test	PTE = Valid	Bit Aligned	0	TB Miss Write Prot	TB Flush Not Modify
1E						
1F						

100	System Init
101	Unaligned Data Trap
102	Page Trap
103	Modify Bit
104	Protection Violation
105	Translation Buffer Miss
106	Reserved Floating Operand
107	Translation Buffer Parity Error
108	Cache Parity Error
109	Reserved
10A	Reserved
10B	Reserved
10C	RDS Error
10D	Timeout
10E	Odd Address Error
10F	Control Store Parity Error

```
(1)   .FIELD Definitions
      These occur at the beginning of the listing, in the source file
UBF.MIC.  They have the form:
      S=MODL/w0,b,c,N
      The first parameter (S) is meaningful only when 'D' is specified
as the default mechanism, and in that case, gives the default value of
the field associated.
      The second parameter (b) gives the size in (decimal) number
of bits.
      The third parameter (c) gives the field position in decimal
as the bit number of the rightmost bit of the field.  Bit zero is defined
here as the LSB.
      The fourth parameter (N) is optional and selects a default
mechanism for the field.  The legal values of this parameter are the
characters "D", or "".
        "D" specifies that the default value of the field is to explicit
        value is specified.
        "" is used on the jump address field to specify that the default
jump address is the address of the next instruction assembled (not.
In general, the symbol "." means "here".
In general, a field is a convenience to the set of bits which provide
some logical function to decoders, or controls for ALUs.
Examples:
      .FIELD/S,5,20,D
      The microcode field named S contains five bits wide
and the rightmost bit is in bit 20 of the 32 bit word as bit 20 of the
microinstruction.  It is set to 'A' specifically requested for any I.N.D.
this field is omitted, it ensures that the field is 0
      A.FIELD/,5
      The field which controls ALU A port source, and is set
at 25.  The fourth parameter is omitted, so the field
is available to the programmer explicitly to value is explicitly
specified for the field, the assembler.

(2)   Value Definitions
      Following a field definition, symbols may be created to that
field to correspond to values of the field.  The form is:
      SYMBOL,N
      The "N" is, in hex, the value of symbol when used in the field.
examples:
      ALU=0,4,25          ; field definition for which following symbols is set
      XPLUS
      ZERO
      Here the symbols "ADD" and "XPLUS" may be used for the "VALUE" field.
In the assembler, the text, writing "A/ADD" means put the value 7
into the 4-bit field starting at bit 25 of the microword.  The symbols
are chosen for mnemonic value, of course, as any legal
identifier would.  Another "ALU/XOR" or "A/XOR" would be the
equivalent of the A and B inputs".  The text, "A,/ZERO" is read as
"ALU produces the sum of A and 0".
      SCR/=0, .25,D       ; field definition for following symbols
      NOPWC
      U.FIELD
      Here the symbols "WCP" and "LOADP" are chosen for the field named
"SCR", which controls the shifting of the register.  To shift write
SCR/WCP in every microinstruction in which we do not want the SC
to change, but to increment it, or use the default mechanism, which
assumes that unless a field instruction explicitly specifies a change
to SC ( e.g. SCR/LOADP), the assembler will make the value of this = 0 0.

(3)   Label Definitions
      A micro-instruction may be identified by a symbol followed by colon
preceding the microinstruction.  The address of the
microinstruction includes the value of the symbol in the field named PC.
example:
      FOO:    A/=CO
      This is a micro-instruction symbol (a jump address) containing
the value "FOO".  It also defines the symbol as the address
of itself.  Therefore, it can also be used by the microprocessor, to mean
loop on itself.

(4)   Comment
      A two colon anywhere on a line causes the rest of the line
to be ignored by the assembler.  This feature is to encourage a of comments.
```

```
(5)     A microinstruction definition
        A word of microcode is defined by specifying a field name,
followed by slash (/), followed by a value.  The value may be a
symbol defined for that field, a hex number, or a decimal number
(hex: string identinguished by the fact that it is terminated by
a period).  Several fields may be specified
in any order, with the corresponding field/value specifications with
commas.  Example:
        MEM/LD,BUS/A,ALU/A+B
        The field named "ALU" is given the value symbol "A+B" (in
this case the mnemonic for the A plus B ALU function), field "BUS" has value
"BUS" and "MEM" field "LD" has value "1".

(6)     Continuation
        The definition of a microinstruction may be continued onto two or
more lines by breaking it after any comma.  In other words, if the
last nonblank, non-comment character on a line is a comma, the
instruction specification is carried over to the following line.
Example:
        ANS/LA,SWB/0,           ;Select LA & 0 as ALU inputs
        ALU/A+B                 ;Select ALU to perform A+B
By convention, a plus (+) and a tilde (~) or hyphen appears between
even instructions, to make it easier for the reader to distinguish
continuation from separate microinstructions.

(7)     Macros
        A macro is a symbol which may be used in place of more field/value
specifications and/or macros.  A macro definition is a line beginning
(no spaces) with a quote (") followed by a quote string of characters
which may have followed by a quoted string of a sequence of any
printable characters:
        "A_IS=0         "ANS/LA,SWB/0,ALU/A+B,D_ALU"
The appearance of "A_IS=0" in a microinstruction definition is equivalent
to the appearance of the string.  Macros may have parameters, enclosed in
square brackets ([ and ]).  The actual parameters given when the macro is
invoked may optionally be preceded by a percent sign (%) to indicate that the given value should be, and
may be followed by a decimal digit string to indicate an octal symbol in
the macro body which is replaced by the parameter:
        "DJ[_,%%]       "JMP/D,MPA/%2,A_IS/%1,D_DEC/%1"
        This macro indicates that its first parameter (optionally %'d)
should be used as the value in an "ASH_DEC" field, and its second parameter
as the value of the "MPA" field.  A typical use of this macro might
look like:
        DJ[I,1$0]
In this case, the macro would be expanded to "JMP/D,MPA/1$0,A_IS/I,D_DEC/1".
Macros can use other macros in their definitions.

(8)     Pseudo Ops
        In addition to the above, the following pseudo operations
.LOCTR and .LOCTR + [no. or "1"] tell the assembler that subsequent microcode will
be loaded into memory locations beginning with the specified address and memory will be
assigned in a sequential manner.
.TITLE defines a string of text to appear in the page header, and
.IDENT defines an entry for the table of contents at the beginning.
.PAGE causes a new page of listing and creates a .TOC entry if
        a .TITLE or .IDENT has been encountered.
.DEL defines the value of a macro parameter if it is not otherwise specified.
.CHANGE redefines a previously defined macro.
.DEFAULT assigns a value to an undefined macro parameter.
.IF causes assembly if the parameter value is zero, and
.IFNOT causes assembly if the parameter value is non-zero.
.ENDIF terminates assembly if suppressed by the parameter named.
.RAM sends the microcode to RAM or the high, or the microinstruction
.MEASURE causes the listing to be sorted by execution count
.DEBUG defines the desired parts of the listing.
.TOC* and .NOCREF enable and disable the collection of cross-reference
        information on symbol usage.
.LIST and .NOLIST enable and disable output listing.
.WIN and .NOWIN enable and disable writing words to the left window
        for a new output.
.WORKFILE selects a new assembler definition needed for specific microprocessors.
```

```
(5)     Location Control
        A 4-word instruction labelled with a number is assigned to that
number.
        The character "#" at the beginning of a line, to mean "
a string of 0's, 1's, and/or 1's, specifies a constraint on the
address of following microinstructions.  The number of characters
in the constraint string (excluding the "#") is the number of low-order
bits constrained in the address.  The microassembler attempts to find
an unused location whose address has 0 bits in the positions
corresponding to 0's in the constraint string, and 1 bits where the
constraint has 1's.  Attempts will be made to avoid bit positions.
        If there are any 0's in the constraint string, the constraint
specifies a number of distinct addresses.  There is the number of 1's
in the string.  All locations in the block will have 1's in the address
bits corresponding to 1's in the string, and all positions marked by #'s
will be the same in all locations in the block.
        In such a multiple block, the default address progression is
counting in the "#" positions of the constraint string.  But a new
constraint string causing a split block may force skipping over
some locations of the block.  Within a block, a new constraint
string does not change the pattern of default address progression, it
merely advances the location counter over those locations.  The
microassembler will allocate blocks
        A bit constraint string ("#") followed by anything but "#",
        or "#" serves to terminate a constraint block.
.swap ab:
        x1
        This specifies that the low-order address bit must be zero.
The microassembler looks at the two-word pair of locations, and puts
the next two word-wide instructions there.
        x11
        This specifies that the two low-order bits of the address must
both be one.  Since there are no 2's in this constraint, the
assembler finds only one location matching the constraint.
        a[1]x[1]
        This specifies a pair of addresses, the first having a zero in
the  bit, and the second having a one in that position, but all
other bit positions the same.
```

```
;FIELDS ARRANGED ALPHABETICALLY

ACF/<0:3:I>             ;ACCELERATOR CONTROL
        NOP=0
        SYNC=1
        TB=2
        CONTROL=3       ;ACCELERATOR-DEPENDENT CONTROL FUNCTION

ACM/<0:2:I>             ;ACCELERATOR MICRO-EMULATION CONTROL
        SYNC.LP=1
        ACCEPT=2        ;RETURN ACCL TO MONITORING CRD
        YOLK.DCK=3

AUX/<0:1:I>             ;AUXILIARY CONTROL
        CBIT
        TB=1
.ENDF                   ;ENABLE DEF OF ALL FUNCTIONS

ALU/<0:4:I>             ;ALU CONTROL FUNCTIONS
        A+B
        A+B TLOC=01
        A+1+B=02
        TEST.DCK=03                     ;INSTRUCTION DEPENDENT
        A-B-1=04
        A-B=05
        A-B TLOC=06
        DPA(B)=07                       ;A .OR. .NOT. B
        C24=10                          ;A .AND. B
        A(B)NOT 08                      ;A .AND. .NOT. B
        NOT(A)=0A                       ;.NOT. A
        A-B/2, C=0B
        0=0C                            ;A .OR. B
        A(B)=0D                         ;A .AND. B
        B=0E
        A=0F
.ENDDEF

AMX/<0:2:I>             ;MUX TO ALU
        =A=0
        B=A=1
        0=MBR.DCK=2                     ;MEMORY BUFFER REGISTER ADDRESSING IS DT
        MBR=3.DCK=3                     ;MEMORY DATA REGISTER. DCK(1:0)

BRN/<0:3:I:"0 1">       ;BRANCH ENABLE
        NO=0                            ;NO BRANCH
        CS=1                            ; ALL 2
        RCR=2                           ;ENTRY, PSL<CM>, LA(30)
        CS1=3                           ; 01: CM1, 0
        ACCE.=4                         ;CASE FROM ACCEL CONTROL
        0010.(Y+F):5                    ;(A) MODE) =, ASPC+ASTE, ASPE+D+D
                                        ;   2 NORMAL, 1  00=D OR DOUBLE
                                        ;   3=FIELD,  3=ADDRESS
        END.OF1 6                       ;(1) MODE) =, D ERROR, 1 CLASS=(D)
        132=1=7                         ;(=3 MODE) =, 1=2(1)
        PC.<0:2>=8                      ;(1) MODE) =, SAVE+ASMST=(A)+D-NST, DST A=PC
        PS1=09                          ;(A) MODE) MODE.(SS.(2)TALL =, =
        SAC M1=0A                       ;( 1) MODE) SRC A=PC
        10, R21.0B                      ; 2=T2 MISS, 1=ERROR
                                        ; 2=STALL, 3=DATA GR
        WL.=0C                          ;SCLK6.0, D(1:0)
        SCC<5=0D                        ;MODE=, (A+B), D=A
        INTERRUPT=0E                    ;=, (=), INTERRUPT  INT PCQ
        DECIMAL=0F                      ;0, 1 BYTE 0 VALID DIGIT, DECIMAL EXCEPTION
        MICROTRAP=10                    ;MICROTRAP DISPATCH VECTOR
        TAG .(0+-11                     ;=FPD, NESTED ERROR, LOW TAG BITS)
                                        ; 3=READ LATERLOCK, 5=TEST, MBOX CMD
                                        ; 2=WRITE, 3=READ, WRITE CMD
        EA.=12                          ;PSL(1 H, PATH 2, NO.MBO.1, 0
        SCM=13                          ;SCLK(1:0).MB.0, SD 07.0, SCASE(1:0).E.C
        # 0'=3=15                       ;RLOG EVENT, PLOK(1:0)=0, ALL 1, ALL=0
                                        ; (ALL DATS FROM PREVIOUS STATE)
        STATE/=3:16                     ;STATE =2(4)
        STATE=3:17                      ;STATE =3(0)
```

```
        0:BYTE=S+-8                     ;BYTES 3, 2, 1, 0 OF D.REG.0
        03-0:IB                         ;INSTR
        PSL.CC*1A                       ;N,Z,V,C OF PSL
        ALU*4                           ;ALU N, ALU Z, CRCEV, ALU COUT
        PSL.MLDG*1C                     ;*MASK*(300, *CURSOR*, 15*M, *KERNEL
        *0.TEST*1D                      ;PTE VALID, ALIGNED, 0---0, *
                                        ; 1--1ACCESS:1,01 04, 1--4M C-B, ANY W,0
                                        ; 0--ACCESS VIOLATION, 3 *TO MODE

BMX/=0 R.R>             ;BMX TO ALU
        MASK 0                          ;4 0 IN THE BIT SELECTED BY %D<1:0>
        SC.DF.LD*1                      ;1 = UNPACK BUFC, 1-BY FD
        PACKED.*1=2                     ;PACKED FLOATING
        LB*3
        ID*4
        MSUM
        <Y3:C
        RBMX*7                          ;0 BM W

DCR/=0,2.3.0            ;CONDITION CODE=
        IID*3                           ;DEFAULT
        LOAD.USCC*1                     ;SAMPLE ALU & ALU CONDITIONS
        SET/=/                          ;FORCE =, NO EFFECT ON N  Z, C
        INT/=1                          ;1 N Z  IF ALU.N=1.U.
                                        ; INT N FROM MAR[HOT]
        R3N.4                           ;SET V & Z FROM =1, C FROM =M IO
        N.Z.1.=*3                       ;SET = AND Z FROM R.U[HOT]
        C.LSB*=6                        ;DT DOES UNAFFECTED
        INST 0-1=1

CSD/=0,*,*/             ;CONSOLE & IO M.R CONTROL IF R3/1
        MOD.*                           ;DEFAULT, ALLOW AUTO IE PC=0
        AC*=E                           ;SET CONSOLE ACKNOWLEDGE F.=G
        CONT=7                          ;CLEAR CONSOLE MODE
        READ SCAN                       ;READ IO BUS TAG SELECTED BY SI
        4R3U.R3/ EQ                     ;READ IO BUS 4BH SELECTED BY R3=B3
        MOIE.=0:EQ                      ;WRITE R3=B PRIMARY BY SL
        MOIE.R4W*0F                     ;W.TA PCA SELECTED BY U=W4

D4/=0,4.05.0
        MD*=1                           ;DEFAULT, MDLC
         ****/=/                        ;DC=SLE SHIFT LEFT
        4:MPY 2                         ;MD=M= SHIFT MIGHT
        0.2.1                           ;-- N.U ALU 0=1, S-IFT LEFT
                                        ; CLSE LOAD FROM 0.F
        LEFT=5                          ;SHIFT LEFT
        TIGHT=E                         ;SHIFT RIGHT
        <=>=4                           ;LOAD S=F MUX, EXPONENT F INS
        <=>=.:=                         ;LOAD S=F MUX, DIVIDEND R.S/1:0 FORMAT
        SD.R.1.0                        ;LOAD SIGNIFICAND DATA FROM D* S.S
        3*FC.SL1*0B                     ;SELECT BYTES AROUND BIT 16
        C=1C                            ;LOAD C THIJ CAL
        SAL.SC=0:                       ;LOAD SAL.SC,
        SAL.=SL=                        ;LOAD SAL'44F SAL'
        <1:4 2F                         ;LOAD <F:1:0>

U/=0,3,78.0             ;DATA TYPE
                        ;SIGNED/UNSIGNED/ZERO EXTENDED, SHF AMOUNT,
                        ;CONDITION CODE SETTING, AND NORMAL RESULT=
         *=1=0=0                        ;DEFAULT
        NORD=1
        0-7E=2
        INST.0=F.1                      ;INSTRUCTION DEPENDENT =
                                        ;MY 0= SHARE OF D=WD/DOUBLE
```

```
ALU/<0:3, 5                             ;EXPONENT ALU
        Add
        Sub
        Clear
        Minuend 1
        Zero
        A-Zero
        B-Zero
        Select
        M<15:0>/= 7                     ; CASE(M<8>)

SBWX/<0:2,5:6           ;SBWX TO CALU
        <Zero>                          ;DEFAULT
        M<8:1>
        Zero:CPP-2
        Shift Value:0                   ;SHIFT VALUE

<PA/<0:1,29:0           ;PE REGISTER CONTROL
        MEM:0                           ;DEFAULT: MEM2
        LED:1

FS/<0:1,42              ;FUNCTION SELECT FOR FS AND MD
        MCT=0                           ;ENABLE MEMORY CONTROL
        LED=1                           ;ENABLE LED DIS AND CONSOLE CONTROL

TCR/<0:2,36             ;INTERRUPT AND EXCEPTION ACKNOWLEDGE
        <0:2>
        STOP=1                          ;STROBE INTERRUPT REQUEST
        INTR=2                          ;INTERRUPT ACKNOWLEDGE
        EXC=4                           ;EXCEPTION ACKNOWLEDGE

CBD/<0:4,33:1           ;SBI CONTROL FUNCTIONS
        <0:2>                           ;DEFAULT
        STOP=1
        <1:0>=2                         ;FLUSH LB AND LOAD SSA
        A_RS=4
        CLR.0:1 4                       ;CLEAR SILO 0,1 (=1) SELECTED
        CLR.2:3 5                       ;CLEAR SILO 2,3 (=1) SELECTED
        RESET=6                         ;TRANSFER ERROR SYNDROME
        CLR.0=07                        ;CLEAR BYTE 0 (WAR SPECIFIED)
        CLR.1=08                        ;CLEAR BYTE 1 (WAR SPECIFIED)
        CLR.0-3.09                      ;CLEAR BYTES 0-3 (=1 IF S DATA)
        CLR.1 5.00M2=0F                 ;CLEAR BYTES 4-7 (SAME AS ABOVE)
                                        ; IF TRUE IN MC SPECIFIED, WORD IN
                                        ; CLEAR SPECIFIED.  IF A SELF CONTAINED
                                        ; SPECIFIED, CLEAR IT.  IF SPECIFIED,
                                        ; ABSOLUTE, OR DISPLACEMENT, CLEAR THE
                                        ; INTERVAL SPECIFIED.

ED.+DDR/<5,6,:4         ;DU BUS CONTROL
        IDLE 0                  ;AD     ;APPLICATION/DIAGNOSTIC DATA TO/FROM DU
        DAY TIME=*              ;AD=A   ;CURRENT TIME OF DAY
                                        ; MUST READ BYTE, STORE SEPARATELY
        RCV 1=3                 ;AD     ;SYSTEM IDENTIFICATION
        RSG= 5                  ;AD=AM  ;CONSOLE RECEIVE CONTROL/STATUS
        RSG1 2                  ;AD     ;CONSOLE RECEIVE DATA BUFFER
                                        ; (1:0) RECEIVED
        TSC=06                  ;RC=AR  ;CONSOLE TRANSMIT CONTROL/STATUS
        TSD=7                   ;AT     ;CONSOLE TRANSMIT DATA BUFFER
                                        ; (1:0)=ID REGISTER
        DQ 1                            ;BASE DATA SYS REGISTERS (MAINT ONLY)
        VAT.BIT 3               ;WA     ;UNIBUS  VECTOR ADDRESS MODIFIER
        CLR.IE=0A               ;RD=AR  ;CNTL=0  CLEAR INTERRUPT ENABLE
        INTERVAL=0B             ;RE     ;CURRENT INTERVAL COUNT
        IP=0C                   ;RD=AR  ;CPU ERROR/STATUS
        WCT.LH 0D               ;RD=WA  ;EXCEPTION & INTERRUPT CONTROL
        EIP=0E                  ;RD=AR  ;UNIBUS  INTERRUPT VECTOR
        PSL=0F                  ;RD=AT  ;PROCESSOR  STATUS LONGWORD
        TB/=10                          ;TRANSLATION BUFFER DATA
        TBSTS=11                        ;TB ERROR/STATUS 0
        TB=TAG=12                       ;TB ERROR/STATUS 1
        ACL.0 14                        ;ACCL PORT M REGISTER #0
        ACC.1=15                        ;ACCL PORT M REGISTER #1
        ACC.2=16                        ;ACCELERATOR PORT REGISTER #2
        ACC.05= 7                       ;ACCELERATOR CONTROL/STATUS
```

```
                SILO=1A                         ;M-BIT 1 OF FROM SBI REGISTER
                SBI=M=a=9                       ;SBI ERROR REGISTER
                TIME=OUT=A                      ;SBI TIMEOUT ADDRESS
                FAULT=ID                        ;FAULT/STATUS
                LOAD=1C                         ;SBI SILO COMPARISON
                MAINT=1D                        ;SBI MAINTENANCE
                PARITY=1E                       ;CACHE PARITY
                MICRO=STACK                     ;MICRO STACK
                MICRO=DATA=1F                   ;MICRO DATA
                WCS.ADDR=20
                WCS.DATA=21             ;WCS    ;WRITING WCS CONTROL MEMORY
                PROC=0=BASE=2=2                 ;PROCESS 0 BASE 2 REGISTER
                PROC=1=BASE=2=3                 ;PROCESS 1 BASE 3 REGISTER
                P0BR=24                         ; P0 BASE REGISTER
                P1BR=25                         ; P1 BASE REGISTER
                SBR=26                          ;SYSTEM BASE REGISTER
                KSP=27                          ;KERNEL STACK POINTER
                ESP=28                          ;EXEC STACK POINTER
                SSP=29                          ;SUPERVISOR STACK POINTER
                USP=2A                          ;USER STACK POINTER
                INT=SP                          ;INTERRUPT STACK POINTER
                R=0=2C
                R=1=2E
                R=2=2F
                T0=30                           ;GENERAL TEMPS
                T1=31
                T2=32
                T3=33
                T4=34
                T5=35
                T6=36
                T7=37
                T8=3E
                T9=3D
                PCB=3A                          ;PROCESS CONTROL BLOCK BASE
                MEM=3B                          ;SYSTEM CONTROL BLOCK BASE
                P0LR=3C                         ;PROCESS 0 LENGTH REGISTER
                P1LR=3D                         ;PROCESS 1 LENGTH REGISTER
                SLR=3E                          ;SYSTEM LENGTH REGISTER

U=0.:X 0.1              ;NEXT MICRO ADDR SHOULD BE, DEFAULT IS THE
                        ;FOLLOWING ADDR %0=0
                        ;SYMBOLIC ADDR DEFINED IN "1"

ADD=F.F.XH              ;CONSTANTS OR F FORM FN
                .=0                             ;M CALM F4
                .=1                             ;M CALM F4
                .=2                             ;M PTAM F4
                .=3                             ;M PTAM F4
                .=4                             ;M PTAM F4
                S=1.0.M.S                       ;M=0 S=1 4 CONSTANT
                S=0.ISM.S                       ;M=1 ISP S CONSTANT (=1: MODE)
                A=RC.C                          ; OR REGS (=A4 MODE)
                SC.S                            ;SCLD 0, FROM FN
                        ;0 - 3F: CONSTANTS (1 CYCLE SHIFT IF ALU IS SHIFT MODE)
                                ;DECIMAL VALUE OF CONSTANT
                .=0=M                           ;00     ;M,0,0,M=1
                .=1=M                           ;01     ;M,1,M=1
                .=0=0=                          ;02     ;M,2
                .=0=05                          ;03     ;M,3
                .=0=07                          ;04     ;M,4,  M=1,1F
                .=0=0=                          ;05     ;M,5,M=1
                .=1=0.0=                        ;06=    ;M,6,
                .=0=09                          ;07=    ;M,7,
                .=1=1C                          ;08     ;M,8,  M=1,FF
                .=2=0=1                         ;09=20  ;M,9,
                .=7=1=                          ;0A     ;M,A,=1
                .=1=0=13                        ;0B     ;M,B,=0=1
                .=1=17                          ;0C     ;M,C,
                .=1=1C                          ;0D     ;M,D,M=1,FF
                .=1=1C                          ;0E     ;M,E,
                .=1=17                          ;0F     ;M,F,
                .=1=18                          ;10     ;M,10,1A,=2
                .=1=1=                          ;11     ;M,11,=0=1=2
                .=1=1=13                        ;12=    ;M,12,=
                .=1=1C                          ;13     ;CM,...TF,M1)
```

```
.FFFF+16                        ;16     (OP,TF,MH)
.20+ D                          ;32     (OP,AL,MH,TF)
.37+16                          ;48     (OP,AP,MH,TF)
.3F+ 0                          ;64     (MH,AP,TF)
.3F+10                           ;(80)  (AP)
.4F+1                           ;16     (OP,AL,TF,MH)
.5F+32                          ;32     (TF)
.7F+22                          ;96     (AP,AL,MH,TF)
.9F+FFFF20                      ;(128)  (AL,MH)
.B0 00                          ;176    (MH)
.D000-B0                        ;       (CM)
.FC+27                          ;(224   (AP)
.FF+FFFF                        ;(255   (AL)
.00 00                          ;(0     (TF)
SPARE 2A
 0FCF+B0                        ;?      (AL)
 FFFF+FC                        ;(-1?   (AP)
.FFFF 30                        ;(-15   (AP)
.1F+3E                          ;(25    (AP)
.FFF3+2F                        ;(-7    (AP)
.FFFF+F0                        ;(-1    (MH,---TF)
.MM 21                          ;(36    (AP)
 3032+32                        ;?      (TF)
 FE+23                          ;(-2    (TF)
.07+10                          ;(80    (TF,MH)
.0 40                           ;0      (OP,AL,TF)
.1+30                           ;?      (OP)
.FFF6+37                        ;(-10   (OP)
.F+FF+6F                        ;(-11   (OP)
.1A 4+                          ;(86    (AP,AP,TF)
.34 3F                          ;(96    (AP A)
.10+35                          ;(37    (OM,AP,TF)
.FF1+3C                         ;(-4    (OM,TF,MH)
.A+30                           ;(10    (AP,MH)
.7E 3C                          ;(-23   (AP,TF)
SPARE 3F
```

```
MCY+35.0.41.0                   ;MEMORY CONTROL
        TEST EC+FAP3                    TEST "BUT WITH READ CHECK
        WAIT N FAP3                     WAITHER CPU MBR IO GETS WITH CYC F
        Inv .1.AP4 3A                   INV "INAP WITH WRITE C ICC
        #RITE.V.+0C3+.EA            ;#RITE, INHIBIT ETAPS
        #RITE.V.+C+E+0C             ;#RITE, DIAGNO #ARI+1F
        IVTE+3.IFE .+EC+E+0E        ;INTERLOCK #P IF, V.#HIN, ADDRESS
        #READ.V.M+E+0D              ;READ, NORMAL VARIETY
        +E+0D.V.3.0+E.1F            ;E+40   INHIBIT TRAPS
        EC+0.V.#C.0 .14             ;E+40 + M MODIFY
        7E+0 V.1300K+E              ;E+40   M+0K #TH#IFIED BY IBUFFER
        7E+0 1.+C.7E+ 0             ;READ, NO IVTE C FOR STALL
                                    ; DATA GOOD  M IVA M IBUFFER A #FET
        .0 +AP3 .+.V0E0+ 3A         ;#MPTLOCK READ, INHIBIT E+E+F
        .0C+AIC.+.+0E+E 1C          ;#MPTLOCK READ, NORMAL VARIETY
        SET.DO.1 30                 ;+40 # C #DC NECESSITY
        SET MCLE.+VALAM+32          ;READ1 3+1
        INVA.LOCK1+2C               ;CLEAR E+E+F #+1#2+F
        V. 0A1+#38                  ;DECREMENTATION IS #ORDE #ALID
        Exter.1+.+ 0A               ;EXTENDED WRITE TO CLEAR 3E1 +#+1#+
        #RITE.+ 2A                  ;#RITE, PHYSICAL
        LOC.#A1E7.A+ 3F             ;+1+BLOCK #RITE, PHYSICAL
        READ.T+32                   ;READ, PHYSICAL
        #+40 IST SLV+D5             ;INTF+.3+ -IMMARY READ
        LOCK+40 P+3A                ;INTERLOCK READ, +#+SICAL
```

```
        M_[1:].IN.READ_SC               ;SI=0 IS A CYCLE IF [1 WANTS 0=0

VEC/ 0.=1 01.0
        %SF 1                           ;SET=ALL
        CMA.CM=01                       ;CREATE NEW PSL FOR CHM
        CMA.[1:] %P= 30                 ;[7:2] IF 4.[:7]%=1, 4[.[1:]]=%=0
        CMA: [...]=VEC=03
        [PL..=4                         ; M = STATE OR INSTRUCTION OPCODE
        LD=B STATE=05
        LD=1 ACC [2=06                  ;TAKE CONDITION CODES FROM ACCELERATOR
        %LAD 4[3]=19                    ;LOAD PCP FLAG STACK[
        T %.P=...=28                    ;CLEAR [%,[=*]0> 4[]
        %P .[%]. 29                     ;SET =SAME
        CLR.[SCT.ERR=08                 ;CLR REGTED ERROR FLAG IN CPU STATUS
        SET.[SCT.E[R=08                 ;SET SAME
        SETSCT.[P=07                    ;SET UNALIGNED DATA REFERENCE
        %P[4]:[0.[]=[ 00                ;4[%[.= SAVED CONTEXT, [%=[0] - [0[4%
        %P 4[.]%AP 0E                   ;4[P.= SAVED CONTEXT IS [N % M=>
        [%=[.P.[LD=3F                   ;4.[0% %>= 0P FULL 32-31> =0[>%32

*[0[/=0,[,[.[.[         ;ADDRESS CLUST CONTROL
        M.= 1                           ;DP=%[0[]
        SC_[.[.]
        SC_[[A=1
        [A[4=0                          ;[A_[A=4
        [C[.=[6                         ;PC_[[.]
        [C[.=[2                         ;PC_[[2]
        [C[.=[4                         ;PC_[[4]
        [C=0. 7                         ;PC_[[=[.,  M IS DETAIL MOD OF INSTR BUFFER

D4/=0,4.[1.C
        %DF=7                           ;DEFAULT, HOLD
        [P[[.=1                         ;LD.SLP SP=[P LEFT 2
        [[SH 2 0                        ;LD.SLP SH=[P R[GHT 2
        LEFT 5
        RIGHT 0
        SHF=C                           ;LOAD SHF, INTEGER FORMAT
        SHF.FL=0                        ;LOAD SHF, UNPACKED FLOATING FORMAT
        ZE[.CO[=38                      ;SELECT CONSTANT - 0/8 IS PATH M[A[.E
                                        ; FOR WHICH ALU CRY OUT IS FALSE
        =[LEL CO                        ;LOAD ACCELERATOR D=T= FROM OF BUS
        C=CC
        CO=0E                           ;LOAD .0 M.E
        GL=0P                           ;LOAD /=50

=AM7/=0,[.[/[.M         ;DATA PATH MEMORY TO AMD
        0=0                             ;DEFAULT
        0=[

TBM/=0,[.[7             ;DATA PATH MEMORY TO TBM.  SAME BIT AS MAME
        M=0
        M=[

SBR/=0,[,[0.0           ;SC REGISTER CONTROL
        MD=0                            ;DEFAULT, HOLD
        LDAI=                           ;LOAD =MBR<09[00

SSM/=0,0,0B.0           ;Slow CONTROLS
        %[0[.0                          ;DEFAULT
        LD.[.SS_1                       ;SS_AL.[ SP
        SS_[RCV.[D=2                    ;SP=ID
        MDT.SS=3                        ;SD=[NT ID
        SD_[NT=1 =3=3                   ;SS_SS
        %% [0[.[[=6                     ;SS_MI.[[SA, SS_SS.WDS.AL.[[0
        PTE.SWI.[                       ;SS_ALU=[SA, SS_SS.WDS.ALU=[SA.WDS..R=[1
        [.P.SB=65.7                     ;CLEAR DCTH

SHF/=0.3 05.0           ;ALU SHIFTER CONTROL
        A_[AF                           ;[=P=A...[, S=P_[A[.[
        [EFT=1                          ;SHF_=...[[1), [S=PRT AT [A[1
        =[SH .3                         ;SHF_=...[P[), INSERT 32 [A[1
        M.....[ 5                       ;SHF_=...[2[ [D.[[.3.L3[, INSERT 0
        %[2[T=0 7                       ;SHF_=...[12), INSERT 32 [NTL
        LEFT=0                          ;SHF 4 ...[[0)
```

```
D1/=0.3.25.0            ;D-FIELD (=A-1 FOR MOST
                                ;       ALU     D       D
                                ;       ---     -       -
        =1=0=1                  ;       PSL+RD  D31     ALU C31
        RS=R-1                  ;       ALU D   D3      D31
        =0=0=0                  ;       0       0       D31
        SCP=0=3                 ;       0       0       S
        SP=PSE=4
        =1=m5                   ;       D31     D31     ALU C31
        PSH=0=H                 ;       0       ALU D,1 S
        PSH=-V                  ;       1       ALU D,1 1

SM4/=0.2.12             ;(CCXPR F' =H
        SALU=0                          ;SALU 0=1CD
        PS=1                            ;P=<PSL>
        RL=R7                           ;RL/<SP'02A
        SLL.RA/=3                       ;ALU/C4 02?

XFU/=0,7,3=.0                   ;SCRATCH PAD U'UTP=, 7 BITS
        MDP=0                           Use=Ut]
        LCAC.LC,SC=6                      CAC LC, ALW SC[30:00]
        WTITE PC'STW=7                  ;WRITE RC, MDP=SC[03+00,

XFU.AA/=0,7,X4          ;4 FUNCTION BITS OF SPC FIELD
        LB.=.LA3.1                      ;CAC LA, .LA PS'W PC[03+]
        LCAC.LA 3                       ;CAC LA_PN, =ALU L3
        WTITE.RAB=2                     ;WRITE RA, SD [=C%]

XFU.ADM/=0,1,16         ;AC NUMBER IN SPC FIELD
                        ;MAX MODE               +4              +4
        S31.SF1=0                       =0      AP1 R           A=1 R
        S32 SF2=1                       =1      SP3 R           S=2 R
        S32 SF1=0                       =2      SP2 R           SP1 R
        P4M=X                           =3      *PM             P=3
        P3M=1 A                         =4      *PM=1           P4M=1
        SC=5                            =5      AIMXEDLUDE>     SL=32:00>
        S31 1=6                         =6      SP1 R-1         SP1 R=1

XFU.ADM1/=0,1,45        ;AC NUMBER IN SPC FIELD == 1] MODE
                        ;=11 MODE               +4              +4
        S30.SFC=0                       =0      =MC M           S=0 R
        DST DST=1                       =1      DST R           DST R
        DST S'G=2                       =2      DST R           S=2 R
        1=MT.=PM=X                      =3      SRC R           S=C R
        S=0.04.' =4                     =4      SRC R .CR. '    S=C R .CR. '
        SC 5                            =5      =CXPR]=05>      =<CAXCEED>

SPD.R/=0.3,29           ;SUTA'=P- PAR FUNCA WITH LDA = BET= OF SP AS ADF
        LCAC.LC=2                       ;.CAC LC, ALW SPD.RM
        WTITE.PC=4                      ;WRITE RC
        LCAC.LA=4                       ;.CAC LA, LB
        WTI'E.R=B=5                     ;WTI'E RA, RB
        LCAC.LA3.WTITE-RC=6               ; CAC LA, LA[=1], AND WRITE RC[RM]
        LCAC.LC.WTITE,PAM'=7            ;.CAC LC[PM], AND WRITE PA. RC[RI]
```

```
SPD,RAV<0,0,06          ;R4/R5 LOCATIONS
        R0:0
        R1:1
        R2:2
        R3:3
        R4:4
        R5:5
        R6:6
        R7:7
        AP=0C                   ;412 = ARGUMENT LIST POINTER
        FP=0D                   ;413 = STACK FRAME POINTER
        SP=0E                   ;414 = STACK POINTER
        R15=F                   ;415 = PC, TO SOFTWARE, SCRATCH TO UCODE
SPD,RC<4,4,0>           ;RC LOCATIONS
        T0:0
        T1:1
        T2:2
        T3:3
        T4:4
        T5=5
        T6=6
        T7:7
        LC.SAVE                 ;RAW SAVE SPACE IS -FFF
        VA.SAVE
        PTE.SAVE
        PTE.MODIFY
        PTE.SAVE
        SC.SAVE
        VA.SET_QE
        MDIF.VA:CF
        PTE.MASK=0F

SJR<4,2,R4,0            ;SUBROUTINE CONTROL
        NOP:0                   ;(DEFAULT)
        CALL:1                  ;PUSH CUR. CP + N.A ONTO CONSTRUCTION
                                ; CNTR STACK
        RET:2                   ; CRF TOP OF STACK TO UPC
                                ; AND POP STACK
        SAVE:3                  ;REPLACE CUR S BITS IN RET
                                ; ADDRESS WHICH IS SAVED FROM
                                ; PREVIOUS CALL

VAL<0,1,75,0
        SUPV:0                  ;DEFAULT
        LOAD:1                  ;LOAD VA

                        ; 1:0   .N.M.   RE ENABLE LISTING SPACE FOR DEBUG OUTPUT
                        ; 1:1   .LOAD   RE ENABLE CROSS PROCESSOR
                        ; 50:0
```

```
;ALU O/P - THRU ALU_D.AND...

ALU_C(A)          'AMX/RAVX,OXT,DT/LONG,ALU/A'
ALU_C[]D          'ALU/#1,ANY/RAMX,OXT,LONG,BVX/REMX,RSVX/D'
ALU_C-B           'AMX/RAVX,OXT,DT/LONG,RBVX/C,BMX/RBMX,ALU/A-B'
ALU_C-B-1         'AMX/RAVX,OXT,DT/LONG,RBVX/C,BMX/RBMX,ALU/A-B+1'
ALU_C-D           'AMX/RAVX,OXT,DT/LONG,RBVX/D,BMX/RBMX,ALU/A-B'
ALU_C-K[]         'AMX/RAVX,OXT,DT/LONG,KMX/#1,BMX/KVX,ALU/A-B'
ALU_C-K[]+'       'KMX/#1,BMX/KVX,AVX/RAMX,OXT,DT/LONG,ALU/A-B+1'
ALU_C-K[]         'KMX/#1,BMX/AVX,AVX/RAMX,OXT,DT/LONG,ALU/A+B'
ALU_C-K[]+'       'KMX/#1,BMX/AVX,AVX/RAMX,OXT,DT/LONG,ALU/A+B+1'
ALU_C-LB-1        'AMX/RAVX,OXT,DT/LONG,BMX/LB,ALU/A+B-1'
ALU_C+LC          'AMX/RAVX,OXT,DT/LONG,BMX/LC,ALU/A+B'
ALU_C-LL          'AMX/RAVX,OXT,DT/LONG,BMX/LC,ALU/A-B'
ALU_C+LC+1        'AMX/RAVX,OXT,DT/LONG,BMX/LC,ALU/A+B-1'
ALU_C-LL-1        'AMX/RAVX,OXT,DT/LONG,BMX/LC,ALU/A-B-1'
ALU_C-MASK+1      'AMX/RAVX,OXT,DT/LONG,'MX/MASK,ALU/A-B+1'
ALU_C-D           'AMX/RAVX,OXT,DT/LONG,RBVX/D,BMX/RBMX,ALU/A-B'
ALU_C-D           'AMX/RAVX,OXT,DT/LONG,RBVX/C,BMX/RBMX,ALU/A-B'
ALU_C-D-1         'AMX/RAVX,OXT,DT/LONG,RBVX/D,BMX/RBMX,ALU/A-B+1'
ALU_C+D+1         'AMX/RAVX,OXT,DT/LONG,RBVX/C,BMX/RBMX,ALU/A+B+1'
ALU_-I            'AVX/RAMX,OXT,DT/LONG,ALU/NOT A'
ALU_D             'RAVX/D,AVX/RAMX,ALU/A'
ALU_D.OXT[]       'RAVX/D,AVX/RAMX,OXT,DT/#1,ALU/A'
ALU_D.OXT[].AND.K[]       'RAMX/D,AVX/RAMX,OXT,DT/#1,KMX/#2,'VX/XVX,ALU/AND'
ALU_D.OXT[].AND.K[]       'ALU/AND/#1,AMX/RAVX,OXT,DT/#1,RAMX/D,BVX/RMX,KMX/#2'
ALU_D.OXT[]+K[]   'RAVX/D,AMX/RAMX,OXT,DT/#1,KMX/#2,CMX/RMX,ALU/A+B'
ALU_D.OXT[]-K[]   'RAVX/D,AMX/RAMX,OXT,DT/#1,KMX/#2,RMX/RMX,ALU/A-B'
ALU_D.OXT[]+LC    'ALU/A+B,AVX/RAMX,OXT,DT/#1,RAVX/D,BMX/LC'
ALU_D.OXT[]-D     'ALU/A+B,AVX/RAMX,OXT,DT/#1,RAVX/D,BMX/RBVX,RBMX/D'
ALU_D.OXT[]-D     'RAMX/D,AM: 'RAVX,OXT,DT/#1,RBMX/D,BMX/RBMX,ALL/A-B'
ALU_D.AND.K[]     'RAMX/D,AMX/RAVX,AVX/#1,BMX/KMX,ALU/AND'
ALU_D.AND.MASK    'RAMX/D,AMX/RAVX,BVX/MASK,ALU/AND'
ALU_D.ANDNOT.K[]          'RAVX/D,AMX/RAMX,KMX/#1,BMX/KMX,ALU/ANDNOT'
ALU_D.ANDNOT.MASK         'RAVX/D,AMX/RAVX,BMX/MASK,ALL/ANDNOT'
ALU_D.ANDNOT.D    'RAMX/D,AM+,RAVX,RBMX/C,BVX/RBMX,ALU/ANDNOT'
```

```
;ALU_D(E);... THIS ALU_D.XOR ...

ALU_3151        =RBMA/D,BMA/R3XX,ALU/3
ALU_D_1K(*)     =RBMA/D,BMA/R4XX,ALU/A+B,BTA/XYA,ALU/A+1
ALU_D_1K(*)     =RBMA/D,BMA/R4XX,ALU/A+B,BTA/XYA,ALU/A-B'
ALU_D+4(*)      =RBMA/D,BMA/R4XX,ALU/A+B,BTA/XYA,ALU/A-B'
ALU_D+8(*)      =RBMA/D,BMA/R4XX,ALU/A+B,BTA/XYA,ALU/A-B'
ALU_D+C(*)+1    =RBMA/D,BMA/R4XX,ALU/A+B,BTA/XYA,ALU/A+B-1'
ALU_D+C(*)-1    =RBMA/D,BMA/R4XX,ALU/A+B,BTA/XYA,ALU/A-B-1'
ALU_D+LB        =RBMA/D,BMA/R4XX,BMX/C,ALU/A-B
ALU_D+LC        =RBMA/D,BMA/R4XX,BMX/C,ALU/A+1
ALU_D+LC        =RBMA/D,BMA/R4XX,BMX/C,ALU/A-B'
ALU_D+LC        =RBMA/D,BMA/R4XX,BMX/C,ALU/A-B-1'
ALU_D+0-PSL.C   =RBMA/D,BMA/R4XX,BMX/C,ALU/A+B-PS_C'
ALU_D+0-PSL.C   =RBMA/D,BMA/R4XX,KNX/#1,BMX/#XX,ALU/OR'
ALU_D.R4X[*]    =RBMA/D,BMA/R4XX,BMX/C,ALU/OR'
ALU_D.R4X=LC    =RBMA/D,BMA/R4XX,BMX/C,BTA/PSW,ALU/ORNOT'
ALU_D.R4X=LC    =RBMA/D,BMA/R4XX,BMX/C,BTA/PSW,ALU/OR'
ALU_D.R4X+0     =RBMA/D,BMA/R4XX,BMX/C,BTA/RBMA,ALU/M1'
ALL_D.JD        =RBMA/D,BMA/R4XX,BMX/C,BTA/RBMA,ALU/A1B'
ALL_D+0         =RBMA/D,BMA/R4XX,BMX/C,BTA/RBMA,ALU/A-B'
ALL_D+0         =RBMA/D,BMA/R4XX,KNX/#1,BMX/#XX,ALU/A-B'
ALL_D+0         =RBMA/D,BMA/R4XX,RBMA/C,BTA/RBMX,ALU/A+B-1'
ALU_D+0+1       =RBMA/D,BMA/R4XX,RBMA/C,BTA/RBMX,ALU/A+1
ALU_D+0-1       =RBMA/D,BMA/R4XX,RBMA/C,BTA/RBMX,ALU/A-1'
ALU_D+0+SEXT(*) =RLU/A-B+PS_C,BMX/=XX,BMA/=BMX,RBMA/C,RBMX/D
ALU_D.SX'[*]    =RMAX/J,BMX/R4XX,SX'_D'/F',ALU/A'
ALU_D.SX'[*]    =RMAX/D,BMA/R4XX,RBMX/SX'_D'/F',KNX/#2,BMX/#XX,ALU/XYA'
ALU_D.SX'[*]+LI =RMAX/J,BMX/R4XX,SX'_D'/F',KNX/#2,BMX/#XX,ALU/A+B'
ALU_D.XD'_R[*]  =RMAX/J,BMX/R4XX,BMX/#1,BMX/#XX,ALU/XD'
ALU_D.XD'_LC    =RMAX/D,BMA/R4XX,BMX/C,ALU/XOR'
ALU_D.XD'_LC    =RMAX/D,BMA/R4XX,BMX/C,BMX/XBYA,ALU/XOR'
ALU_D.XD'_SI/   =RMAX/E,BTA/RBMX,SPD.R/D4C.LC,SPD.RE/#1,BMX/LB,ALU/XOR'
ALU_D.XOR,SI[*] =RMAX/E,BTA/RBMX,SPD.R/D4C.LC,SPD.RE/#1,BMX/LC,ALU/XOR'
```

```
;ALU_K... THRU ALL PC...

ALU_K[]             "RMX/#1,BMX/KVX,ALU/B"
ALU_LA              "RMX/LA,ALU/A"
ALU_LA,ANDLA[]      "ANX/LA,RSX/#1,BVX/KVX,ALU/AND"
ALU_LA,ANDNOT,K[]       "ANX/LA,KVX/#1,BMX/KVX,ALU/ANDNOT"
ALU_LA,ANDNOT,MASK      "ANX/LA,PNX/MASK,ALU/ANDNOT"
ALU_LA,XOR,LC       "AMX/LA,BMX/LC,ALU/XOR"
ALU_LA[]D           "AMX/LA,RBNX/D,BMX/RBVX,ALU/A+"
ALU_LA-D            "AMX/LA,RBNX/D,BMX/RBVX,ALU/A-B"
ALU_LA-D-1          "AMX/LA,RBNX/D,BMX/RBVX,ALU/A-B+1"
ALU_LA+K[]          "AMX/LA,KMX/#1,BMX/KMX,ALU/A+B"
ALU_LA-K[]          "AMX/LA,KMX/#1,BMX/KMX,ALU/A-B"
ALU_LA-K[]+1        "ALU/A+B+1",AMX/LA,BMX/KMX,KVX/#1"
ALU_LA+K[],RLOG     "AMX/LA,KMX/#1,BMX/KMX,ALU/A+B,RLOG"
ALU_LA-K[],RLOG     "AMX/LA,KMX/#1,BMX/KMX,ALU/A-B,RLOG"
ALU_LA[],B          "AMX/LA,PMX/LB,ALU/A"
ALU_LA-LC           "ALU/A-B,AMX/LA,BVX/LC"
ALU_LA[]Q           "AMX/LA,RBNX/Q,BMX/RBVX,ALU/A+"
ALU_LA+Q            "ALU/A+B,AMX/LA,BVX/RBMA,RBVX/Q"
ALU_LA-Q            "ALU/A-B,AMX/LA,BVX/RBMA,RBVX/Q"
ALU_LA-Q-1          "ALU/A-B-1",AMX/LA,BMX/RBVX,RBMA/Q"
ALU_LB              "BMX/LB,ALU/B"
ALU_LC              "BMX/LC,ALU/B"
ALU_ROT,B           "ALU/ROTA,AVX/RAMX,RAM/B"
ALU_ROT,RC[]        "SFO,B/LCAT,LC,SPD,RC/#1,BMX/LC,ANX/RAMA,DAT,DT/LONG,ALU/DROT"
ALU_PACK,FP         "BMX/PACKED.FL,ALU/B"
ALU_PC              "BMX/PC,ALU/B"

;ALU_Q... THRU CACHE_...

ALU_Q               "RAMX/Q,ANX/RAMA,ALU/A"
ALU_Q,DAT[]         "RAMX/Q,ANX/RAMA,DT/0",ALU/A"
ALU_Q,DAT[],ANDMASK,K[]     "ALU/ANDMSK,AVX/RAMX,DAT,DT/#1,RAVA/Q,BVX/KMX,KNX/#2"
ALU_Q,DAT[],OR,K[]          "ALU/OR,AVX/RAMX,DAT,DT/#1,RAVA/Q,BNX/AVX,KMX/#2"
ALU_Q,DAT[],OR,Q            "ALU/OR,AVX/RAMX,DAT,DT/#1,RAVA/Q,BNX/RBMX,RBVX/Q"
ALU_Q,DAT[],+D      "ALU/A+B,AVX/RAMX,DAT,DT/#1,PMX/PBVX,RBNX/Q,RANX/Q"
ALU_Q,DAT[],+D+1    "ALU/A+B+1,ANX/RAVA,DAT,DT/#1,BVX/RBMX,RAVA/Q,RBMX/Q"
ALU_Q,DAT[],+4[]    "ALU/A+B,AVX/RANX,DAT,DT/#1,RAVA/Q,BNX/AVX,AVX/#2"
ALU_Q,DAT[],-4[]    "ALU/A-B,ANX/RAMA,DAT,DT/#1,RAMA/Q,BNX/AVX,AVX/#2"
```

```
ALU_Q.AND.K[]    "RAMX/Q,AMX/RAVX.LVX/2',BMX/KMX,ALU/AND"
ALU_Q.ANDNOT.VASK          'RAVX/Q,AVX/RAMX,BMX/MASK,ALU/ANDNOT'
ALU_Q.ANDNOT.K[]           'RAVX/Q,AVX/RAMX,KMX/#1,BMX/KVX.ALU/ANDNOT'
ALU_Q[B]         "RBMX/Q,BMX/RBVX.L U/B"
ALU_Q[]D         "RAMX/Q,AMX/RAVX.RDMX/D,BMX/RBMX.ALU/#1"
ALU_Q-D          "RAMX/Q,AMX/RAVX,RDMX/D,BMX/RBMX,ALU/A-B"
ALU_Q-D-1        "ALU/A-B-1",AMX/RAVX.RAMX/Q,BMX/RBMX,RBMX/D"
ALU_Q+K[]        "RAMX/Q,AMX/'AVX.KVX/2',BMX/KMX,ALU/A+B"
ALU_Q-K[]        "RAMX/Q,AMX/'AVX.KVX/2',BMX/KMX,ALU/A-B"
ALU_Q+K[]-1      "ALU/A-B1",AMX/RAVX.RAMX/Q,BMX/KMX,KMX/#1"
ALU_Q+LB         "RAMX/Q,AMX/RAVX,BVX/.S,ALU/A-B"
ALU_Q-LB         "RAMX/Q,AMX/RAVX,BVX/.S,ALU/A-B"
ALU_Q+LB+1       "RAMX/Q,AMX/RAVX,BVX/.S,ALU/A-B+1"
ALU_Q+LC         "RAMX/Q,AMX/RAVX,BVX/.C,ALU/A-B"
ALU_Q-LC         "RAMX/Q,AMX/RAVX,BVX/.C,ALU/A-B"
ALU_Q+LC1        "ALU/A-B1",AMX/'AVX.RAMX/Q,BMX/.C'
ALU_Q+MASK       "ALU/A-B,AMX/RAMX,RAVX/Q,BMX/MASK"
ALU_Q-MASK-1     "ALU/A-B-  ,AMX/RAVX.RAMX/Q,BMX/MASK"
ALU_Q.OR.K[]     "RAMX/Q,AMX/RAVX,KVX/#1,BMX/KMX,ALU/OR"
ALU_Q.OR.LC      "RAMX/Q,AMX/RAVX,BMX/.C,ALU/OR"
ALU_Q.ORNOT.K[]  "ALU/ORNOT",AMX/RAVX.RAMX/Q,BMX/KMX,KMX/#1"
ALU_Q.SXT[]      "ALU/A,AMX/RAVX.SXT.DT/#1.RAMX/Q"
ALU_Q.SXT[].ANDNOT.K[]   "ALU/ANDNOT,AMX/RAMX.SX1,RAVX/Q,BMX/KVX.KMX/#2,DT/#1"
ALU_Q.SXT[]+K[]  "RAMX/Q,AVX/RAMX.SX1,DT/#1,AMX/#0.BMX/KMX,A,U/A-B"
ALU_Q.SXT[]+.S   "RAMX/Q,AVX/RAMX.SXT.DT/#1,AMX/LB,ALU/A+B"
ALU_Q.SXT[]+PC   "RAVX/Q,AVX/RAMX.SXT.DT/#1,BMX/PC,ALU/A+B"
ALU_Q.XOR.D      "RAVX/Q,AVX/RAMX,BMX/RBVX,RBMX/D,ALU/XOR"
ALU_Q.XOR.K[]    "RAVX/Q,AVX/RAMX,KMX/#1,BMX/KVX.ALU/XOR"
ALU_Q.XOR.LC     "RAVX/Q,AVX/RAMX,BMX/LC,ALL/XOR"
ALU_Q[]          "SPD.R/LOAD.LAB,S-U.RAB/#1,AMX/LA,ALU/A"
ALU_Q[].AND.K[]  "SPD.R/LOAD.LAB,S-U.RAB/#1,AMX/LA,MMX/#2.BVX/KVX,ALL/AND"
ALU_Q[].ANDNOT.VASK      "SPD.R/LOAD.LAB,SPD.RAB/#1,AVX/.A,BMX/MASK,A,U/ANDNOT"
ALU_Q(DST)       "SPD.AC/LOAD.LAB,SPD.ACX(1)/DST.DT,AVX/.A,A,U/A"
ALU_Q[].OR.K[]   "SPD.R/LOAD.LAB,SPD.RAB/#1,AMX/LA,MMX/#2.BVX/KMX,ALU/OR"
ALU_Q[].ORNOT.K[] "ALU/ORNOT,AMX/LA.BMX/KVX.SPD.R/LOAD.LAB,SPD.R+S/2',KMX/#2"
ALU_Q[].XOR.K[]  "SPD.R/LOAD.LAB,S-U.RAB/#1,AMX/LA.MVX/#2.BVX/KMX,ALU/XOR"
ALU_PC[]         "SPD.R/LOAD.LC,S-U.RC/#1.BMX/LC,ALU/B"
ALU_SC(SC)       "SPD/LEAT.(C.SC,TMX/LC,ALL/B"
ALU_R(SP)+K[].B.OS       "SPD.AC/.OAF.LAC,SPD.AC+/SP1.SP1.AVX/LA.KMX/#1,BMX/KMX.A.U/A+B.B.LOC"
```

```
D_DEL_NDRM      .DR/DEL_SV*
D_DEL_SC        .DR/DEL_SC*
D_D_DX[]        .RDNX/D,.RMX/READY,NOT_DT/SV,.ALU/A,.SHF/ALU,DX/SHF*
D_D_DX[]_.NLD   .READY/D,.ALU/A,.RMX/MS,.RNX/.NL/2,.SHF/ALU,DX/SHF*

:D_DEL... TIBI C_D...

D_CACHE_.NCHK[] .VAR/ND*,.VCT/.READ,V.NCHK,.MSC/01,DX/ND*
D[]_CACHE_.VCHK .VAR/ND*,.VCT/.READ,V.NCHK,.DT/#1,DA/AD*
D[]_CACHE_.V    .VAR/ND*,.VCT/.READ,.D[7:1],DX/.CE*
D[]_CACHE_.NOCHK .VAR/ND*,.VCT/.READ,V.NOCHK,.DT/#1,DX/ND*
D[]_CACHE_L[    .VAR/ND*,.VCT/.DCHREAD,V.NCHK,.DT/#1,DA/AD*
D_CACHE_.V[]    .VAR/ND*,.VCT/.DCACHE,V.NCHK,.MSC/01,DX/ND*
D_DECHR_.VCT_DEP .VAR/ND*,.VCT/.READ,V.LDCHK,.D[7:1],DEP,DE/ND*
D_DECHR_.VCT_DEP .VAR/ND*,.VCT/.READ,V.LDCHK,.D[7:1],DEP,DE/ND*
D[_CACHE_L[.BRCH .VAR/.AOP,.NOT/.READ,V.NCHK,.DT/#1,DA/AOP*
E_CACHE[]       .VAR/.AOP,.NOT/.READ,V.NCHK,.MSC/01,DX/AOP*
E[_CACH*        .*VAR/.AOP,.NOT/.READ,V.NCH*,D*/#1,.CM/NO>*
E_BLANK         .E_*[.20]*
E_(((MARC)      .SHF/ALU,DX/SHIFL*
E_ALL_RIGHT2    .SHF/R,SH-2,DX/SHF*
E_ALL_RIGHT     .SHF/R,SH-1,DX/SHF*
E_ALU_LEFT_D    .SHF/LEFT,DP/SHF*
E_ALU_LEFT2     .SHF/ALU,DT/.LOAD,DX/SHF*
E_ALU_LEFT      .SHF/LEFT_DX/SH*
E_ALU           .SHF/ALU,DX/SHF*
E_DECBIASVAC    .RE/ADDCLA,DP/SVAC*
E_-2            .VAR/.RMX,.VCT_DT/.LOAD,.RSNX/Q,.RMX/.RNX,.ALU/A-2,SHF/ALU_DX/SHF*
E_-1            .VAR/.RMX,.VCT_DT/.LOAD,.RSNX/Q,.RMX/.RNX,.ALU/A-1,SHF/ALU_DX/SHF*
E_-R[]+1        .VAR/.RMX,.VCT_DT/.LOAD,.RMX/91,.RMX/.NNX,.ALU/.VAR-[.5+7:0,DR/SHF*
E_-R[]          .VAR/.RMX,.VCT_DT/.LOAD,.RMX/.VC,.ALU/.VAR1,SHF/ALU,DX/.5+E*
E_-2            .VAR/.RMX,.VCT_DT/.LOAD,.RSNX/Q,.RMX/.RNX,.ALU/A-9,SHF/ALL_DX/SHF*
D_DECBIASVAC    .RE/ADDCLA,DP/SVAC*
D_ALU           .SHF/ALL,DX/SHF*
D_ALU_LEFT_D    .SHF/LEFT_DX/SH*
D_ALU_LEFT      .SHF/LEFT_DX/SH*
D_ALU_LEFT2     .SHF/ALU,DT/.LOAD,DX/SHF*
D_ALU_LEFT_D    .SHF/LEFT,DP/SHF*
D_ALL_RIGHT     .SHF/R,SH-1,DX/SHF*
D_ALL_RIGHT2    .SHF/R,SH-2,DX/SHF*
D_(((MARC)      .SHF/ALU,DX/SHIFL*
E_BLANK         .E_*[.20]*
E[_CACH*        .*VAR/.AOP,.NOT/.READ,V.NCH*,D*/#1,.CM/NO>*
E_CACHE[]       .VAR/.AOP,.NOT/.READ,V.NCHK,.MSC/01,DX/.AOP*
E[_CACHE_L[.BRCH .VAR/.AOP,.NOT/.READ,V.NCHK,.DT/#1,DA/.AOP*
D_DECHR_.VCT_DEP .VAR/ND*,.VCT/.READ,V.LDCHK,.D[7:1],DEP,DE/ND*
D[_CACHE_L[     .VAR/.AOP,.NOT/.READ,V.NCHK,.DT/#1,DX/ND*
D[_CACHE_.V     .VAR/ND*,.VCT/.READ,.D[7:1],DX/.CE*
D_CACHE_.NOCHK  .VAR/ND*,.VCT/.READ,V.NOCHK,.DT/#1,DX/ND*
D[_CACHE_.VCHK  .VAR/ND*,.VCT/.READ,V.NCHK,.DT/#1,DA/AD*
D_CACHE_.NCHK[] .VAR/ND*,.VCT/.READ,V.NCHK,.MSC/01,DX/ND*

:D_DEL... TIBI C_D...

D_DEL_NDRM      .DR/DEL_SV*
D_DEL_SC        .DR/DEL_SC*
D_D_DX[]        .RDNX/D,.RMX/READY,NOT_DT/SV,.ALU/A,.SHF/ALU,DX/SHF*
D_D_DX[]_.NLD   .READY/D,.ALU/A,.RMX/MS,.RNX/.NL/2,.SHF/ALU,DX/SHF*
```

```
D_D.CAT[]+K[]     'K4X/D,K4X/RAMX,CAT,DT/P1,BMX/M2,BNX/4YX,ALU/A+B,SHF/ALU,DK/SHF'
D_D.CAT[].OR.V    'K4X/D,K4X/RAMX,CAT,DT/P1,BMX/T,BNX/PBUS,ALU/OR,SHF/ALU,DK/SHF'
D_D.CAT[]+Q       'ALU/A+B,K4X/RAMX,CAT,DT/P1,BMX/RAMX,BNX/Q,D_A_J'
D_D.CAT[]+D-1     'K4X/D,RNA/RAMX,CAT,DT/P1,BMX/RBYX,ALU/A+B+1,D_ALU'
D_D.CAT[].XOR.Q   'K4X/SHF,ALU/XOR,SHF/ALU,RNX/K4X,0X',BMX/Q,D'/P1,BYX/Q,BNX/RAMX'
D_D.CAT[].XOR.YC[]      'K4XX/D,4YX/RAMX,CAT,DT/P1,SPC.R/LOAD.LC,SPC.RC/P2,RAMX/C,ALU/XOR,SHF/ALU,DK/SHF'
D_D.AND.K[]       'DAMX/D,RNX/RAMX,RMA/P1,BMX/RNX,ALL/AND,SHF/ALU,DK/SHF'
D_D.AND.K[].LEFT2       'K4YX/D,4YX/RAMX,DXX/P1,BYX/4YX,ALU/AND,SHF/ALU,DT,DT/LONG,DK/SHF'
D_D.AND.K[].RIGHT       'K4YX/D,4YX/RAMX,DXX/P1,TYX/4YX,ALU/AND,SHF/RIGHT,DK/SHF'
D_D.AND.,C        'RANX/D,ANX/RAMX,PWX/.C,ALU/AND,SHF/ALU,DK/SHF'
D_D.AND.MASK      'RANX/D,ANX/RAYX,BMX/MASK,ALU/AND,SHF/ALU,DK/SHF'
D_D.AND.Q         'RANX/D,ANX/RAZX,BMX/Q,BMX/BMX,ALU/AND,SHF/ALU,DK/SHF'
D_D.AND.RC[]      'RANX/D,ANX/RAYX,SPD.R/LOAD.LC,SPD.RC/P1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF'
D_D.ANDNOT.K[]    'RANX/D,ANX/RAYX,KYX/2^.BMX/BMX,ALU/ANDNOT,SHF/ALU,DK/SHF'
D_D.ANDNOT.LJ     'RANX/D,ANX/RAYX,BYX/.C,ALU/ANDNOT,SHF/ALU,DK/SHF'
D_D.ANDNOT.-SHZ   'DK/SHF,ALU/ANDNOT,ANX/RAMX,RMXX/3,BMX/RMX,4YX/.C,SHF/ALU'
D_D.ANDNOT.Q      'RANX/D,ANX/RAYX,RBMX/D,BMX/BMX,ALU/ANDNOT,SHF/ALU,DK/SHF'
D_D.ANDNOT.RC[]   'RANX/D,ANX/RAYX,SPD.R/LOAD.LC,SPD.RC/P1,BMX/LC,ALU/ANDNOT,SHF/ALU,DK/SHF'
D_D(RAC)          'RANX/D,ANX/RAYX,ALU/A,SHF/ALU,DK/SHF,FL'
D_D[.K[]          'RANX/D,ANX/RAYX,KYX/2C,BYX/BMX,ALU/M1,SHF/ALL,DK/SHF'
D_D-K[]           'RANX/D,ANX/RAYX,4YX/2^,BYX/BMX,ALU/A+B,SHF/ALU,DK/SHF'
D_D-K.]           'RANX/D,ANX/RAYX,4YX/2^,BYX/BMX,ALU/A-B,SHF/ALU,DK/SHF'
D_D-K[]+1         'RANX/D,ANX/RAYX,4YX/2^,BYX/BMX,ALU/A+B-1,SHF/ALL,DK/SHF'
D_D-LB            'RANX/D,ANX/RAYX,BYX/LB,ALU/A+B,SHF/ALU,DK/SHF'
D_D-LC            'RANX/D,ANX/RAYX,BYX/LC,ALU/A+B,SHF/ALU,DK/SHF'
D_D-LC            'RANX/D,ANX/RAYX,BYX/LC,ALU/A+B,SHF/ALU,DK/SHF'
D_D-LC+PS..C      'RANX/D,ANX/RAYX,BYX/LC,ALU/A+BIPSL.C,SHF/ALU,DK/SHF'
D_D.LEFT          'DK/LE^'
D_D.LEFT2         'DK/LE^ 2'
D_D[]?A+K         'RANX/D,ANX/RANX,BYX/NASK,ALU/M1,SHF/ALL,DK/SHF'
D_D.OR.ASCII      'D_D.OR.4^.70]'
D_D.OR.4^         'RANX/D,ANX/RANX,4YX/01,BYX/RDR,ALU/OR,SHF/ALU,DK/SHF'
D_D.ORNOT.NASK    'RANX/D,ANX/RAYX,BYX/MASK,ALU/ORNOT,SHF/ALU,DK/SHF'
D_D.OR.PSL2       'DK/SHF,ALL/L^,ANX/RAMX,RANX/D,BMX/BYX,BMX/.1,SHF/ALU'
D_D.OR.PSAV       'DK/SHF,ALU OR,ANX/RAYX,RANX/D,BMX/4YX,BMX/.2,SHF/ALU'
D_D.OR.Q          'RANX/D,ANX/RANX,BYX/.C,TYX/REMY,ALU/OR,SHF/ALU,DK/SHF'
D_D.OR.RC[]       'RANX/D,ANX/RANX,SPD.R/LOAD.LC,SPD.RC/P1,BYX/LC,ALU/OR,SHF/ALU,DK/SHF'
D_D[]?            'K4X/D,4YX/RANX,+B.R/2,BYX/BMX,ALU/M1,SHF/ALL,DK/SHF'
D_D+Q             'K4X/D,4YX/RANX,+B.R/2,BYX/RBMX,ALU/A+B,SHF/ALU,DK/SHF'
D_D-Q             'K4X/D,4YX/RANX,+B.R/2,BYX/RBMX,ALU/A-B,SHF/ALU,DK/SHF'
```

```
D_D+Q+1         'AVA/C,AVX/BANX,FDVA/C,DVX/REMX,ALU/A+B+1,SHF/ALU,DX/SHF'
D_D-Q-1         'AVA/C,AVX/BANX,FDVX/C,DVX/REMX,ALU/A-B-1,SHF/ALU,DX/SHF'
D_D.TIGHT       'DA/RIGHT'
D_D.TIGHT2      'DA/RIGHT*2'
D_D.TIGHT[3]    'FDVX/C,DVX/BNX,ALU/B SHF/RIGHT,DA/SHF'
D_D.UNPK        'DA/SXTB SKAP'
D_D.XAT[]       'AVA/C,AVX/BANX,EXT,DT/#1,ALU/A,SHF/ALU,DX/SHF'
D_D.XXT[].RIGHT 'AVA/C,AVX/BANX,EXT,DY/#1,ALU/A,SHF/RIGHT,DA/SHF'
D_D.XOR.A[]     'AVA/C,AVX/BANX,BNX/#1,BNX/AVX,ALU/XOR,SHF/ALU,DX/SHF'
D_D.XOR.LO      'AVX/C,AVX/BANX,BNX/LO,ALU/XOR,SHF/ALU,DX/SHF'
D_D.XOR.Q       'AVA/C,ANX/BANX,BNX/Q,BNX/AVX,ALU/XOR,SHF/ALU,DX/SHF'

;D_INT.SUB...  THRU D_TC...

D_INT.SUB       'MOV/FRAD,INT SUB,DA/MAP'
D_X[]           'BNX/#1,BNX/AVX ALU/B,SHF/ALU,DX/SHF'
D_X[].RIGHT     'BNX/#1,BNX/AVX ALU/B,SHF/RIGHT,DA/SHF'
D_X[].RIGHT2    'BNX/#1,BNX/AVX,ALU/B,SHF/RIGHT2,DA/SHF'
D_LA            'AVA/LA,ALU/A,SHF/ALU,DA/SHF'
D_LA.AND.M[]    'AVA/LA,BNX/#1,BNX/BNX,ALU/AND,SHF/ALU,DA/SHF'
D_LAID+PSL.C    'AVA/LA,BNX/3,BNX/BAVX,ALU/A+B+PSL.C,SHF/ALU,DA/SHF'
D_LA+D          'DX/SHF,ALU/A+B,AVX/LA,DVX/REMX,BAVX/D,SHF/ALU'
D_LA[]*RAC      'ANX/LA,ALU/A,SHF/ALU,DA/SHF,FL'
D_LA-M[]        'AVA/LA,BNX/#1,BNX/BNX,ALU/A-B,SHF/ALU,DA/SHF'
D_LA.RIGHT      'AVA/LA,ALU/A,SHF/RIGHT,DX/SHF'
D_LB            'BVA/LB,ALU/B,SHF/ALU,DA/SHF'
D_LB.PC         'BVA/PC OR LB,ALU/B,SHF/ALU,DA/SHF'
D_LC            'BVA/LC,ALU/B,SHF/ALU,DA/SHF'
D_LC[FRAC]      'BVA/LC,ALU/B,SHF/ALU,DA/SHF,FL'
D_NOT.D         'AVA/D,ANX/BANX,ALL/NOTA,SHF/ALU,DX/SHF'
D_NOT.M[]       'AVA/#1,BNX/BNX,ANX/BAVX,DAT,D'/LONG,ALU/ORNOT,SHF/ALU,DA/SHF'
D_NOT.MASK      'DVA/MASK,BNX/BANX,DAT,D1'/BYTE,ALU/ORNOT,SHF/ALU,DA/SHF'
D_NOT.Q         'AVA/Q,ANX/BANX,ALL/NOTA,SHF/ALU,DX/SHF'
D_NOT.R[]       'LA_DA/#1,AVA/LA,ALU/NOT'D_ALU'
D_PACK.FP       'SVX/PACKED,FL,ALU/B,SHF/ALU,DX/SHF'
D_PACK.FP.LEFT  'SVX/PACKED,FL,ALU/B,SHF/LEFT,DA/SHF'
D_PC            'BVX/PC,ALL/B,SHF/ALL,DX/SHF'
D_PC.LEFT       'PCX/PC,ALL/B,SHF/LEFT,DX/SHF'
```

```
;B_1...                 'DN,SR'
0_0                     .
0_3,0KY[]               RAM:0,ANX:RAMK.OR.,D'/A1,ALU/A.SHF/A,U,DN/SHF'
0_3,AXB.=L)             'RAM:0,ANX:RAMK,RMA,SR1,RMC:RMA,ALL/AND.SHF/ALU,DX/SHF'
0_3,AXB.LC              'RAM:0,ANX:RAMK,RMA,SR1,RMC:RMA,ALL/AND.SHF/ALU,DX/SHF'
0_3,AXB.NASK            'RAM:0,ANX:RAMK,RMA,SR1,RMC:RMA,ALL/AND.SHF/ALU,DX/SHF'
0_3,AXBNOT.)            RAM:0,ANX:RAMK,RMA,SR1,RMC:RMA,ALU/ANDNOT.SHF/ALU,DX/SHF'
0_3,AXBNOT_A[]          'RAM:0,ANX:RAMK,RMA,SR1,RMC:RMA,ALU/ANDNOT.SHF/ALU,DX/SHF'
0_3,AXBNOT_NASK         'RAM:0,ANX:RAMK,RMA,RMASK,ALU/ANDNOT.SHF/ALU,DX/SHF'
0_3,AXBNOT_PSVC         'DX/SHF.ALU/ANDNOT,ANX/RAMK,YATA,B.BMX/RMK,RMA/.1,SHF/ALU'
0_3,AXBNOT_PSVN         'DX/SHF.ALU/ANDNOT,ANX/RAMK,RMA:0.BMX/RMK,RMA/.0,SHF/ALU'
0_3,AXBNOT_PSWP         'DX/SHF.ALU/ANDNOT,ANX/RAMK,RMA:0.BMX/RMK,RMA/.2,SHF/ALU'
0_0,AXB.RC[]            'RAM:0,ANX:RAMK,RMA,SR1,RMC:RMA,ALU/ANDNOT.SHF/ALU,DX/SHF'
3E5_310                 'RAM:0,ANX:RAMK,RMA,SR1,RMC:RMA,ALU/A+B.SHF/A,U,DX/SHF'
0_0=a                   'RAM:0,ANX:RAMK,RMA,SR1,B'.BMX/REMA,ALU/A+B.SHF/A,U,DX/SHF'
0_0=3                   'RAM:0,ANX:RAMK,RMA,SR1,B'.BMX/REMA,ALU/A-B.SHF/A,U,DX/SHF'
0_0=0-1                 'RAM:0,ANX:RAMK,RMA,SR1,B'.BMX/REMA,ALU/A-B.SHF/A,U,DX/SHF'
0_0[]0                  'RAM:0,ANX:RAMK,RMA,SR1,B'.BMX/REMA,ALU/A-B-1.SHF/A,U,DX/SHF'
0_0=RSAG)               'RAM:0,ANX:RAMK,RMA,SR1,B'.BMX/RSMK,ALU/W1.SHF/ALU,DX/SHF'
0_0=R[]                 'RAM:0,ANX:RAMK,RMA/S,ALU/A.SHF/A1.B,SR'.1.'
0_0=K[]                 'RAM:0,ANX:RAMK,RMA/S,ALU/A.SHF/A1.B,SR'.1.'
0_0=K[]-                'RAM:0,ANX:RAMK,RMA/S,ALU/A.SHF/A1.B,SR'.1.'
0_0=K[]-1               'RAM:0,ANX:RAMK,RMA/S,ALU/A.SHF/A1.B,SR'.1.'
0_0=3                   'RAM:0,ANX:RAMK,RMA/S,ALU/A.SHF/A1.B,SR'.1.'
0_0[]MASK               'RAM:0,ANX:RAMK,RMA,RMASK,ALU/W1.SHF/A,U,DN/SHF'
0_0,1EFF                'RAM:0,ANX:RAMK,RMA,ALU/A.SHF/.EFT.DX/SHF'
0_0,05.KLJ              'RAM:0,ANX:RAMK,RMA,RMASK,ALU/W1,BMX/RMA.ALL/OR.Sh-/A,U,D-/SHF'
0_0,0340_MASK           'RAM:0,ANX:RAMK,RMA,RMASK,A,U/SHIFT.SH-/A,U,DX/SHF'
0_0,04.PSAC             'D-/SHF,ALU/TR.ANX:RAMK,RMA:0,ANX/RAMK,KMX/.1.SHF/ALU'
0_0,05.RC[]             'RAM:0,ANX:RAMK,SR1,B:1040.LC,SPC:RC/W1,BMX/LC,ALU/SR.SHF/ALU,DX/SHF'
0_1                     'RAM:0,ANX:RAMK,BMX/SR1,B:1040.LC,SPC:RC/W1,BMX/LC,ALU/A+B.SHF/A,U,DN/SHF'
0_2-.ISP                'RAM:0,ANX:RAMK,BMX/S,YSC/RC4E.B.DS,ALU/A+B.SHF/ALU,DX/SHF'
0_3,1SCI                'RAM:0,ANX:RAMK,ALU/A.SHF/RC4E.SH-,DN/SHF'
0_3,1SSIC2              'RAM:0,ANX:RAMK,ALU/A.SHF/ALIGH-2,DX/SHF'
0_3,SXT[]               'RAM:0,ANX:RAMK,SR1,B1/W1,ALU/A.SHF/ALU,DX/SHF'
0_5,2SR.RC[]            'RAM:0,ANX:RAMK,SPC:W/040.LC,SH0.RC/W1,BMX/LC,ALU/XOR.SHF/ALU,DN/SHF'
```

```
;D_R... THRU DAVA_...

D_R[]               "SPO.R/LOAD.LAB,SPO.RAS/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_R[].AND.K[]       "SPO.R/LOAD.LAB,SPO.RAS/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_R[].ORNOT.R[]     "LAB_R[@1],AMX/LA,BMX/RMX,RMX/@2,ALU/ORNOT.D_ALU"
D_RC[]              "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SHF/ALU,DK/SHF"
D_RC[]_PC           "BMX/PC,ALU/B,SHF/ALU,DK/SHF,SPO.R/WRITE.RC,SPO.RC/@1"
D_RC(SC)            "SPO/LOAD.LC.SC,BMX/LC,ALU/B,SHF/ALU,DK/SHF"
D_R[](FRAC)         "SPO.R/LOAD.LAS,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF,FL"
D_RLOG              "BMX/Q,VAS/READ.RLOG,ALU/B,SHF/ALU,DK/SHF"
D_RLOG.RIGHT        "BMX/Q,VAS/READ.RLOG,ALU/B,SHF/RIGHT,DK/SHF"
D_R(PRN+1)          "SPO.AC/LOAD.LAB,SPO.ACN/PRN+1,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_R(SC)             "SPO.AC/LOAD.LAB,SPO.ACN/SC,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_R(SP1+1)          "SPO.AC/LOAD.LAB,SPO.ACN/SP1+1,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
DAVA_ALU            "VAK/LOAD,SHF/ALU,DK/SHF"
DAVA_D-K[]          "RAMX/D,AMX/RAMX,BMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,SHF/ALU,DK/SHF"
DAVA_D+LC           "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,DK/SHF"
DAVA_DIQ            "D_DIQ,VAK/LOAD"
DAVA_LA             "AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,DK/SHF"
DAVA_LB             "BMX/LB,ALU/B,VAK/LOAD,SHF/ALU,DK/SHF"
DAVA_Q              "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,DK/Q"
DAVA_Q+LB.PC        "RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,DK/SHF"

;EALU_... THRU FE_...

EALU_D(EXP)         "RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B"
EALU_FE             "EBMX/FE,EALU/B"
EALU_K[]            "KMX/@1,EBMX/KMX,EALU/B"
EALU_R[](EXP)       "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,EBMX/AMX.EXP,EALU/B"
EALU_SC             "EALU/A"
EALU_SC.ANDNOT.K[]          "KMX/@1,EBMX/KMX,EALU/ANDNOT"
EALU_SC+FE          "EBMX/FE,EALU/A+B"
EALU_SC-FE          "EBMX/FE,EALU/A-B"
EALU_SC+K[]         "KMX/@1,EBMX/KMX,EALU/A+B"
EALU_SC-K[]         "KMX/@1,EBMX/KMX,EALU/A-B"
EALU_STATE          "EALU/A,VSC/LOAD.STATE"

FE_D(A)             "AMX/RAMX.EXT,DT/LONG,EBMX/AMX.EXP,EALU/B,FEK/LOAD"
FE_D(EXP)           "RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,FEK/LOAD"
FE_EALU             "FEK/LOAD"
```

```
FE_K[]               "KMX/@1,EBMX/KMX,EALU/B,FEK/LOAD"
FE_LA(EXP)           "AMX/LA,EBMX/AMX.EXP,EALU/B,FEK/LOAD"
FE_MABS(SC-LA(EXP)) "AMX/LA,EBMX/AMX.EXP,EALU/MABS.A-B,FEK/LOAD"
FE_Q(EXP)            "RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,FEK/LOAD"
FE_R[](EXP)          "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,EBMX/AMX.EXP,EALU/B,FEK/LOAD"
FE_SC                "EALU/A,FEK/LOAD"
FE_SC.ANDNOT.FE              "EBMX/FE,EALU/ANDNOT,FEK/LOAD"
FE_SC.ANDNOT.K[]             "KMX/@1,EBMX/KMX,EALU/ANDNOT,FEK/LOAD"
FE_SC+1              "EALU/A+1,FEK/LOAD"
FE_SC+FE             "EBMX/FE,EALU/A+B,FEK/LOAD"
FE_SC-FE             "EBMX/FE,EALU/A-B,FEK/LOAD"
FE&SC_K[]            "KMX/@1,EBMX/KMX,EALU/B,FEK/LOAD,SMX/EALU,SCK/LOAD"
FE_SC+K[]            "KMX/@1,EBMX/KMX,EALU/A+B,FEK/LOAD"
FE_SC-K[]            "KMX/@1,EBMX/KMX,EALU/A-B,FEK/LOAD"
FE_SC+LA(EXP)        "AMX/LA,EBMX/AMX.EXP,EALU/A+B,FEK/LOAD"
FE_SC-LA(EXP)        "AMX/LA,EBMX/AMX.EXP,EALU/A-B,FEK/LOAD"
FE_SC.OR.K[]         "EALU/OR,EBMX/KMX,KMX/@1,FEK/LOAD"
FE_SC-SHF.VAL        "EBMX/SHF.VAL,EALU/A-B,FEK/LOAD"
FE_SHF.VAL           "EBMX/SHF.VAL,EALU/B,FEK/LOAD"

;ID_... THRU LC_...

ID[]_D               "CID/WRITE.RMX,ID.ADDR/@1"
ID_DAND.SYNC         "CID/WRITE.RMX,ADS/IBA,KMX/SP1.CON"
ID_D.SYNC            "CID/WRITE.RMX,ADS/IBA,KMX/SP1.CON,ACF/SYNC"
ID(SC)_D             "CID/WRITE.SC"

K[]                  "KMX/@1"

LA_RA[]              "SPO.AC/LOAD.LA,SPO.RAB/@1"
LA_R(DST)SLB_R(SRC)        "SPO.AC/LOAD.LAB,SPO.ACN/DST.SRC"
LA_R(SP2)SLB_R(SP1)        "SPO.AC/LOAD.LAB,SPO.ACN/SP2.SP1"
LAB_R[]&RC[]_G             "ALU_G,LAB_R[]&RC[@1]_ALU"
LAB_R[]&RC[]_ALU "SPO.R/LOAD.LAB,WRITE.RC,SPO.RC/@1,SHF/ALU"
LAB_R[]&RC[]_ALU.RIGHT2    "SPO.R/LOAD.LAB,WRITE.RC,SPO.RC/@1,SHF/RIGHT2"
LAB_R[]&RC[]_D.OXT[]+R[]   "ALU_D.OXT[@2]+R[@3],LAB_R[]&RC[@1]_ALU"
LAB_R[]&RC[]_D+LC          "ALU_D+LC,LAB_R[]&RC[@1]_ALU"
LAB_R[]&RC[]_D-K[]         "ALU_D-K[@2],LAB_R[]&RC[@1]_ALU"
LAB_R[]&RC[]_C-D "SPO.R/LOAD.LAB,WRITE.RC,SPO.RC/@1,ALU/A-B,AMX/RAMX.DXT,DT/LONG,RMX/RBMX,RBMX/D,SHF/ALU"
LAB_R[]&RC[]_D+LC-1        "ALU/A+B-1,AMX/RAMX.DXT,DT/LONG,BMX/LC,SPO.R/LOAD.LAB,WRITE.RC,SPO.RC/@1,SHF/ALU"
```

```
LD3_0[]         "SPU,R:LLEJ,LHE,SPL,PAS/2 "
LD3_R[]         "SPU,R:LLEJ,LHE,SPL,SPO,PCNT /DST-
LD3_R[SP]       "SPO,R:LLEJ,LHE,SPU,PCNT,SP1-
LD3_R[SP+]      "SPO,R:LLEJ,LHE,SPU,PCNT,SP1+
LD3_R[SP-]      "SPO,R:LLEJ,LHE,SPU,PCNT,SP1-
LD_SC[]         "SPU,R:LLEJ,LHE,SPC SC,PA1"
LD_SC[#3:0]     "ALU_L:[#3:0]R1_ALU"
LD_SC[#31:16-#11]   "ALU_H:[#2:16-#11]R1_ALU"
LD_SC[#31:5]    SPU,R:LLEJ,LHE,SPC SC"
LD_SC[#31_5_0]  SPU,R:LLEJ,LHE,SPC SC,SPO,SP1,SP2,SP3,SP4,SP5,SP6,SP7,SP8,SP9,SP10,SP11,SP12,SP13,SP14,SP15,SP16,SP17,SP18,SP19,SP20,SP21,SP22,SP23,SP24,SP25,SP26,SP27,SP28,SP29,SP30,SP31,SPU,R:LLEJ,LHE,SPC SC,PA1"
LD_SC[#31:5-LE]_LE1 "#32/_A_EWE/_3:ALU/A-B_SHF:LEFT,S33,S/_DAD,LC,AR:TE,3+01,SPU,RC/#1"
LD_SC[#31:5-LE]_LE1 "#32/_A_EWE/_3:ALU/A-B_SHF:LEFT,S33,S/_DAD,LC,AR:TF,3+01,SPU,RC/#1"
LD_SC[#31:5-LE]_LE1 "SPU,R:LLEJ,LHE,SPC,RC/#1,S4/ALU,ALU/A+B,4X4/LA,0TB/PAR,PAR/#2"
LD_RC[#31_LE]   "ALL LD,LC,RC[#1]R1_ALU"
LD_RC[#31_0]    "SPC,R:LLEJ,LHE,SPC,RC/#1,S4/ALU,ALU/A+B,4X4/LA,0TB/PAR,PAR/#2"

;PC_... TNSL /EXAS...

PC_PC+1         "PCN/PC+1"
PC_PC+2         "PCN/PC+2"
PC_PC+4         "PCN/PC+4"
PC_PC+8         "PCN/PC+8"
PC_D+PC         "L:U/A-B,XAR:LCAD,PCN/PC_VA,BMX/*0,AMX/R#31,PAR/Q"
PC_VA           "PCN/PC_VA"
PCWAV_A_D       "VA:/_DAD,PCN/PC_VA"
PCWAV_3         "DPR:3,ANX/BMX,A-U/B,XMX:LOAD,PCN/PC_VA"
PCWAV_3_DST[]   "BMX:3,ANX/BMX,A+_D:/PC,ALU/A,XMX:LOAD,PCN/PC_VA"
PCWAV_0_DST[]+PC    "BMX:0,ANX/BMX,A+_D:/PC,ALU/A,XMX:LOAD,PCN/PC_VA"
PCWAV_0_DST[]+PC    "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0_DST[]+PC    "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0_DST[]+PC    "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0-[]      "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0-[]      "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0-PC      "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_B[]       "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_PC        "BMX:0,ALU:B,XMX:LOAD,PCN/PC_VA"
PCWAV_0         "BMX:0,ANX/BMX,A+_D:/PC,XMX:LOAD,PCN/PC_VA"
PCWAV_0-3       "BMX:0,ANX/BMX,A+_D:/PC,SMX/BMX,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0-4[]     "BMX:0,ANX/BMX,A+_D:/PC,SMX/BMX,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0-PC      "BMX:0,ANX/BMX,A+_D:/PC,SMX/BMX,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0-PC      "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0-PC      "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0_SAT[-#0     "BMX:0,ANX/BMX,A+_D:/PC,ALU/A-B,XMX:LOAD,PCN/PC_VA"
PCWAV_0[] XWDWRT,RL:  "SPU,R:LLEJ,LHE,SPC,RC/#1,S4/ALU,ALU/A+B,4X4/LA,0TB/PAR,PAR/#2,BMX/#0,AMX/R#31,ALU/A,DMX:,XMX/LOAD,PCN/PC_VA"
PCWAV_RC[]      "SPU,R:LLEJ,LHE,SPC,RC/#1,S4/ALU,ALU/A+B,4X4/LA,0TB/PAR,PAR/#2,BMX/#0,AMX/R#31,ALU/A,DMX:,XMX/LOAD,PCN/PC_VA"
PC_VIRA         "PC_VIRA"
```

```
;0_0... THRU 0_3...

0_0             0K/CLR"
0_0-0           "WR/RAMA,0ST,3T/LONG,RENX/0,EMX/RBTX,ALU/A+0,SHF/A,U,3K/SHF"
0_0-1           "WR/RAMA,0ST,3T/LONG,RMX/01,EMX/RMX,ALU/A-B,SHF/ALU,0K/SHF"
0_0-LC          "WR/RAMA,0ST,3T/LONG,EMX/LC,ALU/A-B,SHF/A,U,0K/SHF"
0_0+MASK-1      "WR/RAMA,0ST,3T/LONG,EMX/MASK,ALU/A+B1,SHF/ALU,0K/SHF"
0_0-0/LOG       "WR/RAMA,0ST,3T/LONG,RMX/01,ALU/AXB,LOG,SHF/ALU,3K/SHF"
0_0-Y           "WR/RAMA,0ST,3T/LONG,REMX/0,EMX/ZERO,ALU/A+0,SHF/A,U,0A/SHF"
0_ACCF.ASYNC    "0K/ACCEL,PCP/SYNC"
0_ALU           "SHF/A,U,3K/SHF
0_ALU..SFT      "SHF/LEFT,0A/SHF"
0_ALU..SFT2     "SHF/A,U,2*,0A/SHF"
0_ALU.RIGHT     "SHF/RIGHT,0K/SHF"
0_ALU.RIGHT.2   "SHF/RIGHT2,0A/SHF"
0_ALU/=0        "SHF/A,U,0K/SHF,FL"
0_ALU=0         "0K/0"
0_BIFRAC!!31    "RAMA/0,WR/RSVX,ALL/B,SHF/ALL,3K/SHF,FL"
0_B.3XT[]       "RAMA/0,WR/RSVX,3XI,BF/01,ALL/A,SHF/A,U,3K/SHF"
0_B.3XT[,+/-].LEFT "RAMA/0,WR/RSVX,3XI,BF/01,ALU/A+B,SHF/LEFT,3K/SHF"
0_B.4HC.0[]     "RAMA/0,WR/RSVX,3XI,RMX/RSVX,ALU/AXB,SHF/ALU,0A/SHF"
0_B.4HC.0[].RIGHT "R4MA/0,RMX/5HX,XMX/01,3MX/HMX,ALU/AND,SHF/RIGHT,0K/SHF"
0_B.4HC.0[].RIGHT.2 "RAMA/0,RMX/5HX,XMX/01,3MX/HMX,ALU/AND,SHF/RIGHT2,0K/SHF"
0_B.4HC.0[].RCP "RAMA/0,WR/RSVX,SPD.R/LOAD,.C.5P3,RC/01,EMX/LC,ALU/4HC4CT,SHF/ALU,0A/SHF"
0_B.4HEN01.RCP  "RAMA/0,WR/RSVX,SPD.R/LOAD,.C.5P3,RC/01,EMX/LC,ALU/AND,SHF/ALU,0K/SHF"
0_B.AND.RCP     "RAMA/0,WR/RSVX,SPD.R/LOAD,.C.5P3,RC/01,EMX/LC,ALU/AND,SHF/ALU,0K/SHF"
0_BEG.CON       "0K/BEG.CON"
0_B+[]          "RAMA/0,WR/R4YA,4YX/01,.RMX/RMA,ALU/A+B,SHF/ALU,0A/SHF"
0_B-[]          "RAMA/0,WR/R4YA,4YX/01,.RMX/RMA,ALU/A+B,SHF/LEFT,0K/SHF"
0_B-[,+].LEFT   "RAMA/0,WR/R4YA,4YX/01,.RMX/RMA,ALU/A+B,SHF/LEFT,0K/SHF"
0_B-[].         "RAMA/0,WR/R4YA,4YX/01,.RMX/RMA,ALU/A-B,SHF/ALU,0A/SHF"
0_B=LC          "RAMA/0,WR/R4YA,BYX/LC,ALU/A-B,SHF/ALU,0K/SHF"
0_B=LC          "RAMA/0,WR/R4YA,BYX/LC,A.U/A-B,SHF/ALU,3K/SHF"
0_B.LEFT3       "R4YA/0,WR/RSVX,ALU/A,SHF/LEFT3,0K/SHF"
0_B.R3.RII      "R4YA/0,WR/RSVX,RMX/01,3MX/4YA,ALU/2H,SHF/ALU,0A/SHF"
0_B.R3.RII[]    "R4YA/0,WR/RSVX,RMX/01,3MX/4YA,ALU/2H,SHF/ALU,0A/SHF"
0_B.R3.RII[].   "R4YA/0,WR/RSVX,SPD.R/DATA,SPD.RE/01,3MX/LC,ALU/D2,SHF/ALL,0K/SHF"
0_J-0           "R4YA/0,WR/RSVX,RBTX/3,3MX/RSVX,ALU/A-B,SHF/A,U,0K/SHF"
0_J.RIGHT       "R4YA/0,WR/RSVX,ALU/A,SHF/RIGHT,0K/SHF"
0_J.RIGHT2      "R4YA/0,WR/RSVX,ALU/A,SHF/RIGHT2,0K/SHF"
0_J.RIGHT2[]    "R4YA/0,WR/RSVX,ALU/A,SHF/RIGHT2,0K/SHF"
0_J.SXT[]       "RAYA/0,WR/RSVX,3XI,BF/01,ALL/A,SHF/ALU,0K/SHF"
0_J.SXT[,+]     "0K/SHF-A.L,XOR,RMX/4YA,HAM/3,EMX/R4YA,REMX/0,SHF/ALL"
```

```
;Q_IB... THRU Q_PC...

Q_IB.BDEST        "IBC/BDEST,QK/IB,MCT/ALLOW.IB.READ"
Q_IB.DATA         "QK/ID,MCT/ALLOW.IB.READ"
Q_ID[]            "SID/READ.RNA.ID.ADDR/@1,QK/ID"
Q_ID(SC)          "SID/READ.SC,QK/ID"
Q_R[]             "AXA/@1,BMX/RMX,ALU/B,SHF/ALU,QK/SHF"
Q_R[].CTX         "RMX/@1,BMX/RMX.ALU/B,SHF/ALU.DT.DT/INST.DEP,QK/SHF"
Q_R[].RIGHT       "RMX/@1,BMX/RMX.ALU/B,SHF/RIGHT,QK/SHF"
Q_R[].RIGHT2      "RMX/@1,BMX/RMX.ALU/B,SHF/RIGHT2,QK/SHF"
Q_LA              "AMX/LA,ALU/A,SHF/ALU,QK/SHF"
Q_LA.AND.R[]      "AMX/LA,RMX/@1,BMX/RMX.ALU/AND,SHF/ALU,QK/SHF"
Q_LA.ANDNOT.RC[]          "AMX/LA,SPO.R/LOAD.LC.SPO,RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_LA+R[]          "AMX/LA,RMX/@1,BMX/RMX.ALU/A+B,SHF/ALU,QK/SHF"
Q_LA-R[]          "AMX/LA,RMX/@1,BMX/RMX.ALU/A-B,SHF/ALU,QK/SHF"
Q_LA+Q            "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_LB              "BMX/LB,ALU/B,SHF/ALU,QK/SHF"
Q_LC              "BMX/LC,ALU/B,SHF/ALU,QK/SHF"
Q_NOT.Q           "RAMX/Q,AMX/RAMX,ALU/NOTA,SHF/ALU,QK/SHF"
Q_NOT.R[]         "LA_R[](@1),AMX/LA,ALU/NOTA,Q_ALU"
Q_PACK.FP         "BMX/PACKED.FL,ALU/B,SHF/ALU,QK/SHF"
Q_PC              "BMX/PC,ALU/B,SHF/ALU,QK/SHF"

;Q_Q... THRU Q2VA_...

Q_Q.OXT[]-K.]     "RAMX/Q,AMX/RAMX.DXT.DT/@1,KMX/@2,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q.OXT[].LEFT    "RAMX/Q,AMX/RAMX.DXT.DT/@1,ALU/A,SHF/LEFT,QK/SHF"
Q_Q.OXT[].OR.D    "RAMX/Q,AMX/RAMX.DXT.DT/@1,RBMX/D,BMX/RBMX,ALU/OR,SHF/ALU,QK/SHF"
Q_Q.AND.R[]       "RAMX/Q,AMX/RAMX,RMX/@1,BMX/RMX,ALU/AND,SHF/ALU,QK/SHF"
Q_Q.AND.R[].RIGHT         "RAMX/Q,AMX/RAMX,RMX/@1,BMX/RMX,ALU/AND,SHF/RIGHT,QK/SHF"
Q_Q.ANDNOT.D      "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_Q.ANDNOT.R[]    "RAMX/Q,AMX/RAMX,RMX/@1,BMX/RMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_Q.AND.RC[]      "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,QK/SHF"
Q_Q-D             "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q+D             "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q-D-1           "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q(FRAC)         "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,QK/SHF.FL"
Q_Q(FRAC)(B)      "RBMX/Q,BMX/RBMX,ALU/B,SHF/ALU,QK/SHF.FL"
Q_Q+R[]           "RAMX/Q,AMX/RAMX,RMX/@1,BMX/RMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q-R[]           "RAMX/Q,AMX/RAMX,RMX/@1,BMX/RMX,ALU/A-B,SHF/ALU,QK/SHF"
```

```
Q_Q-R[ ]-1          "RAVX/Q,AVX/RANX,BMX/#1,BVX/AVX,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q+LC              "RAMX/Q,AVX/RAMX,BNX/LC,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q-LC              "RAMX/Q,AVX/RAMX,BNX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q-LC-1            "RAMX/Q,AVX/RAMX,BNX/LC,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q.LEFT            "QK/LEFT"

Q_Q-MASK-1          "RAMX/Q,AMX/RAMX,BVX/MASK,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q.OR.K[ ]         "RAMX/Q,AMX/RAMX,AVX/#1,BVX/KMX,ALU/OR,SHF/ALU,QK/SHF"
Q_Q.ORNOT.MASK      "RAMX/Q,AMX/RAMX,BVX/MASK,ALU/ORNOT,SHF/ALU,QK/SHF"
Q_Q+PC              "RAMX/Q,AMX/RAMX,BVX/PC,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q.RIGHT           "QK/RIGHT"
Q_Q.RIGHT2          "QK/RIGHT2"
Q_Q.SXT[ ]          "RAMX/Q,AMX/RAMX,SXT,DT/#1,ALU/A,SHF/ALU,QK/SHF"
Q_R[ ]              "SPO.R/LOAD.LAB,SPO.RAB/#1,AVX/LA,ALU/A,SHF/ALU,QK/SHF"
Q_R[ ].AND.K[ ]     "SPO.R/LOAD.LAB,SPO.RAB/#1,AVX/LA,AVX/XS,BVX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_R[ ].AND.K[ ].RIGHT     "SPO.R/LOAD.LAB,SPO.RAB/#1,AMX/LA,ALU/AND,BMX/KMX,KMX/#2,SHF/RIGHT,QK/SHF"
Q_R[ ].ANDNOT.K[ ]        "SPO.R/LOAD.LAB,SPO.RAB/#1,AMX/LA,KMX/#2,BVX/KMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_RC[ ]             "SPO.R/LOAD.LC,SPO.RC/#1,BMX/LC,ALU/B,SHF/ALU,QK/SHF"
Q_RC[ ].FRAC[ ]     "SPO.R/LOAD.LC,SPO.RC/#1,BMX/LC,ALU/B,SHF/ALU,QK/SHF.FL"
Q_R[ ].FRAC[ ]      "SPO.R/LOAD.LAB,SPO.RAB/#1,AVX/LA,ALU/A,SHF/ALU,QK/SHF.FL"
Q_R(PRN).ANDNOT.Q         "SPO.AC/LOAD.LAB,SPO.ACN/PRN,AVX/LA,BMX/Q,BVX/RBMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_R(PRN+1)          "SPO.AC/LOAD.LAB,SPO.ACN/PRN+1,AMX/LA,ALU/A,SHF/ALU,QK/SHF"
Q_R(PRN+1).AND.Q          "SPO.AC/LOAD.LAB,SPO.ACN/PRN+1,AMX/LA,RBMX/Q,BMX/RBMX,ALU/AND,SHF/ALU,QK/SHF"
Q_R(SHC+1).AND.K[ ]       "SPO.AC/LOAD.LAB,SPO.ACN/1/SRC,OR.1,AMX/LA,KMX/#1,BVX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_SC                "ALU/B,BVX/KMX,KMX/SC,SHF/ALU,QK/SHF"
Q&VA_ALU            "VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_D              "RAMX/D,AMX/RAVX,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_D+LC           "RAMX/D,AMX/RAVX,BVX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_LA             "AVX/LA,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_Q+LB.PC        "RAMX/Q,AMX/RAVX,BVX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF"

;R[ ]_D...THRU R[ ]_PACK.FP

R[ ]_D              "SPO.R/WRITE.RAB,SPO.RAB/#1,AMX/RAVX,OXT,DT/LONG,ALU/A,SHF/ALU"
R[ ]_D-B            "AMX/RAVX,OXT,DT/LONG,RBMX/D,BVX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[ ]_D-R[ ]         "AMX/RAMX,OXT,DT/LONG,AVX/XS,BVX/BMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[ ]_D-LB           "AMX/RAMX,OXT,DT/LONG,BVX/LB,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[ ]_D+LB+1         "AVX/RAMX,OXT,DT/LONG,BVX/LB,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[ ]_D-Q            "AVX/RAMX,OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[ ]_D-1            "AVX/RAMX,OXT,DT/LONG,BMX/KMX,KMX/.1,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[ ]_ALU            "SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
```

```
R[]_ALU.LEFT        "SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/LEFT"
R[]_ALU.RIGHT       "SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_D               "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,ALU/A,SHF/ALU"
R[]_D.AND.K[]       "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/AND,AMX/RAMX,RAMX/D,BMX/KMX,KMX/@2,SHF/ALU"
R[]_D+K[]           "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/A+B,SHF/ALU"
R[]_D-K[]           "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU"
RC_D+K[].RLOG       "SPO.R/WRITE.RAB,SPO.RAB/RC,RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,RLOG,SHF/ALU"
R[]_D-LC-1          "ALU_D-LC-1,R[@1]_ALU"
R[]_D.OR.LC         "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/OR,AMX/RAMX,RAMX/D,BMX/LC,SHF/ALU"
R[]_D.OR.PACK.FP            "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/OR,AMX/RAMX,RAMX/D,BMX/PACKED.FL,SHF/ALU"
R[]_D.OR.Q           "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU"
R[]_D+Q             "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU"
R[]_D-Q             "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU"
R[]_D+Q+1           "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1,SHF/ALU"
R[]_K[]             "BMX/KMX,KMX/@2,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA              "SPO.R/WRITE.RAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU"
R[]_LA.AND.K[]      "AMX/LA,BMX/KMX,KMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA+D            "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA+D+1          "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA-D            "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA+K[]          "AMX/LA,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA+K[]+1        "AMX/LA,BMX/KMX,KMX/@2,ALU/A+B+1,R[@1]_ALU"

R[]_LA-K[]          "AMX/LA,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA+K[].RLOG     "AMX/LA,BMX/KMX,KMX/@2,ALU/A+B,RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA-K[].RLOG     "AMX/LA,BMX/KMX,KMX/@2,ALU/A-B,RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
RC_LA+K[].RLOG      "AMX/LA,BMX/KMX,KMX/@1,ALU/A+B,RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/RC"
RC_LA-K[].RLOG      "AMX/LA,BMX/KMX,KMX/@1,ALU/A-B,RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/RC"
R[]_LA+LC           "AMX/LA,BMX/LC,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA+MASK+1       "AMX/LA,BMX/MASK,ALU/A+B+1,R[@1]_ALU"
R[]_LA-MASK-1       "ALU/A-B-1,AMX/LA,BMX/MASK,SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/ALU"
R[]_LA.OR.D         "AMX/LA,RBMX/D,BMX/RBMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA.ANDNOT.MASK          "AMX/LA,BMX/MASK,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA+Q            "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA-Q            "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LB              "BMX/LB,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LC              "BMX/LC,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LC.RIGHT        "BMX/LC,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_NOT.C           "AMX/RAMX,RAMX/D,DT/LONG,ALU/NOTA,R[@1]_ALU"
R[]_NOT.D           "RAMX/D,AMX/RAMX,ALU/NOTA,R[@1]_ALU"
R[]_NOT.MASK        "BMX/MASK,AMX/RAMX,RAMX/D,DT/LONG,ALU/ORNOT,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_NOT.Q           "RAMX/Q,AMX/RAMX,ALU/NOTA,R[@1]_ALU"
R[]_PACK.FP         "BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
```

```
;R[]_Q... THRU R[]_RLOG...
R[]_Q           "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMA/Q,AYA/RAMX,ALU/A,SHF/ALU"
R[]_Q-1         "ALU_Q+Q-1,R[]_ALU"
R[]_Q+S         "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/A+B+1,BYA/KMX,KMX/.4,AMX/RAMX,RAYA/Q,SHF/ALU"
R[]_Q-S         "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AYA/RAMX,RBMX/S,BMX/RBYX,ALU/A-B,SHF/ALU"
R[]_Q+D+1       "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/A+B+1,AMX/RAMX,RAMA/Q,BMX/RBMX,RBMA/D,SHF/ALU"
R[]_Q+K[]       "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAMX,BYA/KMX,KMX/#2,ALU/A+B,SHF/ALU"
R[]_Q-K[]       "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAMX,BMX/KMX,KMX/#2,ALU/A-B,SHF/ALU"
R[]_Q-K[],RLOG  "RAMX/Q,AYA/RAYX,BYX/KYX,KMX/#2,ALU/A-B,RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE,RAB,SPO.RAB/#1"
R[]_Q+LB        "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/A+B,AMX/RAMX,BMX/LB,RAYA/Q,SHF/ALU"
R[]_Q+LC        "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAYA,BMX/LC,ALU/A+B,SHF/ALU"
R[]_Q-LC        "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAYA,BMX/LC,ALU/A-B,SHF/ALU"
R[]_Q.AND.K[]   "ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/#1,AYX/RAMX,RAYA/Q,BYX/KMX,KMX/#2"
R[]_Q.ANDNOT.K[]        "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/ANDNOT,AMX/RAYA,RAMX/Q,BMX/KMX,KYA/#2,SHF/ALU"
R[]_Q.OR.D      "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/OR,AMX/RAMX,RAMX/Q,BYX/RBMX,RBYA/D,SHF/ALU"
R[]_Q.ORNOT.K[] "SPO.R/WRITE.RAB,SPO.RAB/#1,RAYA/Q,AMX/RAMX,BMX/KYX,KMX/#2,ALU/ORNOT,SHF/ALU"
R[]_Q.RIGHT.1   "ALU_Q,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_RLOG.RIGHT.1        "BYX/BUNSC/READ.RLOG,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/#1"

;RC[]_0... THRU RC[]_D...

RC[]_0          "AMX/RAMX,CAT,DT/LONG,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_D-D        "AMX/RAMX,CAT,DT/LONG,RBMX/D,BMX/RBYX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0+K[]+1    "AMX/RAMX,CAT,DT/LONG,KMX/#2,BMX/KMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0+LC+1     "AYX/RAMX,CAT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0+MASK+1   "AYX/RAMX,CAT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0+MASK+1,RIGHT2    "AMX/RAMX,CAT,DT/LONG,BYA/MASK,ALU/A+B+1,SHF/RIGHT2,SPO.R/WRITE,RC,SPO.RC/#1"
RC[]_ALU        "SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_ALU.LEFT   "SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_ALU.LEFT3  "SPO.R/WRITE.RC,SPO.RC/#1,SHF/ALU,DT,DT/LONG"
RC[]_ALU.LEFT3  "SPO.R/WRITE.RC,SPO.RC/#1,SHF/LEFT3"
RC[]_ALU.RIGHT  "SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_D          "RAMA/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_D.CAT[]    "RAYA/D,AMX/RAMX,CAT,DT/#2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_D.AND.K[]  "RAYA/D,AYX/RAMX,BMX/KYX,KMX/#2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_D.AND.MASK "RAYA/D,AYX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,SPO.R/WRITE,RC,SPO.RC/#1"
RC[]_D.ANDNOT.Q "RAMA/D,AYX/RAMX,RBYA/Q,BYA/RBMX,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_DIBI       "RBMA/D,BMX/RBYX,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_DATA       "RAMA/D,AMX/RAYX,ALU/A,SHF/ALU,DT,DT/IBSY,DFP,SPO.R/WRITE.RC,SPO.RC/#1"
```

```
RC[]_D-A*[]       "RAMX/D,AMX/RAYX,BYX/RMX,RNX/#2,ALU/A+B,SHF/ALU,SPC.R/WRITE,RC,SPD.RC/#1"
RC[]_D-A_1        "RAMX/D,AMX/RAYX,BYX/RMX,RNX/#2,ALU/A+B,SHF/ALU,SPC.R/WRITE,RC,SPD.RC/#1"
RC[]_D.LEFT       "RAMX/D,AMX/RAYX,ALU/A,SHF/LEFT,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_D.LEFT3      "RAMX/D,AMX/RAYX,ALU/A,SHF/LEFT3,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_D.LOG.K[]    "RAMX/D,AMX/RAYX,<YX/#2,BYX/RMX,ALU/OR,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_D.TR.2       "RAMX/D,AMX/RAYX,REMX/D,BYX/R#MX,ALU/OR,SHF/ALU,SPC.R/WRITE,RC,SPD.RC/#1"
RC[]_D.ORNOT.K[]          SPD.RC/#1,SPC.<A*[]E.RC,ALU/EQV[],AMX/RAYX,RANX/D,BNX/RNX,RNX/#2,SHF/ALU"
RC[]_D.SXT[]      "RAMX/D,AMX/RAYX,SXT,DT/#2,ALU/A,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"

;RC[]_X... THRU RC[]RYX_...

RC[]_A*[]         "RMX/#2,BNX/RYX,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_A[]-1        "AMX/RAYX,DST,DT/LCMP,RNX/#2,#WA/AYA,ALU/A+B1',SHF/ALU,SPC.R/WRITE,RC,$PD.RC/#1"
RC[]_A[].LEFT2    "AMX/#2,BNX/AYX,ALU/B,SHF/ALU,DT/LRG,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_A[].RIGHT2   "AMX/#2,BNX/AYX,ALU/B,SHF/RIGHT2,SPD.R/WRITE,RC,SPC.RC/#1"
RC[]_LA           "AMX/LA,ALU/A,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_LA.ANC.K[]           "ALU_LA.AND.K[#2],PC[#1]' ALU"
RC[]_LA.DTX       "AMX/LA,ALU/A,SHF/ALU,DT/DTNST.DEF,SPD.R/WRITE,RC,SPC.RC/#1"
RC[]_LA-B[]       "AMX/LA,RNX/#2,BNX/AYX,ALU/A-B,SHF/ALU,SPC.R/WRITE,RC,SPD.RC/#1"
RC[]_LB           "BMX/LB,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_LB.LEFT      "BMX/LB,ALU/B,SHF/LEFT,SPC.R/WRITE,RC,SPD.RC/#1"
RC[]_LD           "BMX/LC,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_NOT.Q        "RANX/Q,AMX/RAYX,ALU/NOTA,PC[#1]' ALU"
RC[]_PACK.FF      "BMX/PACKFF,FL,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_PC           "BMX/PC,A,U/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q            "RAMX/Q,AMX/RAYX,ALU/A,SHF/ALU,SPD.R/WRITE,RC,SPC.RC/#1"
RC[]_Q.DA*[]      "RAMX/Q,AMX/RAYX,DST,DT/#2,ALU/A,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q.AND.K[]    "RAMX/Q,AMX/RAYX,DTX/RYX,BMX/#2,ALU/AND,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q.ANDNOT.K[]         "RAYX,Q,AMX/RAYX,BMX/RNX,ANX/#2,ALU/ANDNOT,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q-1          "A_U_Q-Q+1,PC[#1]_ALU"
PC[]_Q-A*[]       "RAMX/Q,AMX/RAYX,BYX/RYX,RMX/#2,ALU/A-B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q-A*[]       "RAMX/Q,AYX/RAYX,BYX/AYX,RMX/#2,ALU/A-B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q.LEFT       "RAMX/Q,AYX/RAYX,ALU/A,SHF/LEFT,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q.LEFT3      "RAMX/Q,AYX/RANX,ALU/A,SHF/LEFT3,SPD.R/WRITE,RC,SPD,RC/#1"
PC[]_Q-NASA+1     "RAMX/Q,AYX/RANX,BMX/NASA,ALU/A B-1,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q+PC         "RAMX/Q,AYX/RANX,BMX/PC,ALU/A+B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
PC[]_Q+PC+1       "RAMX/Q,AYX/RANX,BMX/PC,ALU/A+B+1,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q.RIGHT      "RAYX/Q,AYX/RANX,ALU/A,SHF/RIGHT,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_Q.SXT[]      "RAYX/Q,AYX/RANX,SXT,DT/#2,ALU/A,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]_R.LOG.RIGHT  "RMX/Q,WSE/READ.HIGH,ALU/B,SHF/RIGHT,SPC.R/WRITE,RC,SPD.RC/#1"
RC[]AVA_Q+Q       "RAYX/Q,AYX/RANX,RAYX/Q,BMX/RAMA,ALU/A+B,VAR/LGAD,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1"
RC[]SC[]_ALU      "SHF/ALU,SPD/WRITE,RC-SC"
```

```
;R[DST]_...   THRU R[]AAA_...

R[DST]_ALU          "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST"
R[DST]_D            "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST"
R[DST]_D.SHF[].RIGHT     "RAMX/D,AMX/RAMX,SXT,DT/@1,ALU/A,SHF/RIGHT,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST"
R[PRN]_ALU          "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R[PRN]_D            "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R[PRN]_D.OR.Q       "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,R[PRN]_ALU"
R[PRN]_D[]Q         "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1,R[PRN]_ALU"
R[PRN]_D+K[].RLOG  "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R[PRN]_ALU"
R[PRN]_D-K[].RLOG  "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B.RLOG,DT/LONG,R[PRN]_ALU"
R[PRN]_K[]          "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R[PRN]_LA+K[].RLOG        "AMX/LA,KMX/@1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R[PRN]_ALU"
R[PRN]_LA-K[].RLOG        "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B.RLOG,DT/LONG,R[PRN]_ALU"
R[PRN]_LA[]MASK    "AMX/LA,SMX/MASK,ALU/@1,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R[PRN]_LA+Q         "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R[PRN]_PACK.FP      "BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R[PRN]_Q            "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R[PRN+1]_D          "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1
R[PRN+1]_D.OR.Q     "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1"
R[PRN-1]_LC         "BMX/LC,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN-1"
R[PRN-1]_Q          "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN-1"
R[SC]_ALU           "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
R[SC]_D             "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
R[SC]_LA+D          "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
R[SC]_LA-D          "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
R[SC]_LC                       ALU/B,R[SC]_ALU"
R[SC]_Q             "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
R[SP1+1]_LC         "BMX/LC,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1+1"
R[SP1+1]_Q          "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1+1"
R[SP1]_ALU          "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1
R[SP1]_D            "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
R[SP1]_K[]          "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
R[SP1]_PACK.FP      "BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
R[SP1]_Q            "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
R[SRC]_ALU          "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
R[SRC]_D            "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
R[SRC]_D(B)         "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
R[SRC]_D+K[].RLOG  "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,DT/WORD,R[SRC]_ALU"
R[SRC]_D-K[].RLOG  "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B.RLOG,DT/WORD,R[SRC]_ALU"
```

```
P[SIGN] LD          "SYS/LC ALU/B,B]SPD,_ALU"
P[SA2]_D            "<<MR/D,SMX/RAVX ALU/A,SHF/ALL,SPD.WC/WRITE,RAB,SPD.ADR:1/SRC.SRC"
P[SA2]_D_ALU        "SMX/ALU,SPD.WC/WRITE,RAB,SPD.ADR:1/SRC.DR:1"
P[SA2]_D_D[E]       "<<MR/D,RAV<RAVX ALU/B,SHF/ALL,SPD.WC/WRITE,RAB,SPD.ADR:1/SRC.DR:1"
C[RAV_L<1:0>]       "RAV/D,-RAV/SP,BMX/RMX,ALU/A+B,VAR/LOAD,SHF/ALU,SPD.R/WRITE,RAB,SPD.RAB/#1"
C[RAV_L<1:0>]       "RAV/D,-RAV/SP,BMX/RMX,ALU/A+B,VAR/LOAD,SHF/ALU,SPD.R/WRITE,RAB,SPD.RAB/#1"
P[.RAV_L<1:0>_RLPA  "RAV/D,-RAV/SP,BMX/RMX,ALU/A+B,R.LOG.DT/LONG,VAR/LOAD,SHF/ALU,SPD.R/WRITE,RAB,SPD.RAB/#1"
P[.RAV_D<8:0>]      "<<RAV/D,<<RAV/D,BNX/<<RAX/#7,WBX/<<R,ALU/A+B,VAR/LOAD,SPD.R/WRITE,RAB,SPD.RAB/#1"

;SC_...

SC_D(A)             "RMR/RAVX.DXT.DT/LONG,BNX/RVA.EXP,EALU/B,SMX/EALU,SCA/LOAD"
SC_D-M[]            "BMX/RMX,RNX/#1,RVX/RAVX.DXT,D"/LONG,ALU/A-B,SMX/ALU.SCR/LOAD"
SC_ALU              "SMX/ALU,SCR/LOAD"
SC_ALU(EXP)         "SVX/ALU.EXP,SCR/LOAD"
SC_D                "RAMR/D.ADR/RAVX.ALL:4,SMX/ALL.SCR/LOAD"
SC_D.CRY[]-M[]      "RAMR/D,AMR/RAVX.DXT,DT/#1,<<R/#2.BVX/RMX,ALU/A-B.SVX/ALU.SCA/LOAD"
SC_D.DXT,/ZOR.X[]       "RAMR/D,ANX/RAVX.DXT.DT/#1.BVX/RMX,RNX/#2.ALU/XOR,SC_ALU"
SC_D.AND.B[]        "RAMR/D.AMR/RAVX.ALL:4/#1.BVX/RMX,ALU/AND,SMX/ALU,SCA/LOAD"
SC_D(EXP)           "RAMR/D.<<R/RAVX.ALL:4,SMX/ALL.EXP,SCR/LOAD"
SC_D(EXP)[A]        "RAMR/D.<<R/RAVX,CMRX/<<R.EXP,EALU/B,SVX/EALU.SCR/LOAD"
SC_D(EXP)[S]        "PCMR/D,BMR/R2MX.ALU/B.SMX/ALU.EXP,SCR/LOAD"
SC_D-K[]            "RAVX/D,ANX/RAMX,BMX/#1,EMX/RNX,ALU/A+B,SMR/ALU,SCR/LOAD"
SC_D.OR.K[]         "RAVX/D,ANX/RAMX,KMX/#1,BMX/RNX,ALU/OR.SVX/ALU,SCR/LOAD"
SC_D.SXT[]          "RAVX/D,ANX/RAMX.SXT,DT/#1,ALU/A.SMX/ALU,SCR/LOAD"
SC_EALU             "SMX/EALU,SCR/LOAD"
SC_FE               "SMX/FE,SCR/LOAD"
SC_(FB2|SC_FC)      "ESMX/FE.EALU/EABS.K-E.SVX/EALU,SCR/LOAD"
SC_K[]              "KMX/#1,SVX/KMX,EALU/B,SMX/EALL.SCR/LOAD"
SC_K[].ALU          "KMX/#1,RNX/<<R.A L/B,SMX/ALU.SCR/LOAD"
SC_LA               "AMX/.A,ALU/A,SNX/ALU,SCR/LOAD"
SC_LA.AND.B[]       "SMX/.A,KMX/#1,BMX/<<R.ALU/AND,SMX/ALU,SCR/LOAD"
SC_LC(EXP)          "EMX/LC.ALU/B,SMX/ALU.EXP,SCR/LOAD"
SC_PSLADD:          "SMX/EALL.EBNX/RNX,SCR/LOAD,RNX/.F.EALL/B"
SC_Q                "RANX/Q.ANX/<<R.ALU/A SVX/ALU,SCA/LOAD"
SC_Q.AND.A[]        "RANX/Q,ANX/<<R.BVX/RMX,RNX/#1,ALU/AND.SVX/ALU,SCR/LOAD"
SC_Q(EXP)           "RANX/Q,ANX/<<R.EBNX/RNX.EXP,EALU/B.SVX/EALU,SCR/LOAD"
SC_Q(EXP)[B]        "RBNX/Q,ENX/PBV<.A L/B.SVX/ALU.EXP,SCA/LOAD"
SC_Q+M[]            "RANX/Q,AMX/RAV<.BV<KMX,RNX/#1,ALU/A+B.SVX/ALU,SCR/LOAD"
SC_Q-M[]            "RANX/Q,AMX/RAV<.BV<KMX,RNX/#1,ALU/A-B.SVX/ALU,SCR/LOAD"
```

```
SC_Q.XOR.K[]             "RAMX/Q,AMX/RAMX,SVX/KVX,KMX/#1,ALU/XOR,SMX/ALU,SCK/LOAD"
SC_Q.SXT[]               "RAMX/Q,AMX/RAMX.SXT,DT/#1,ALU/A,SMX/ALU,SCK/LOAD"
SC_R[]                   "SPO.R/LOAD.LAB,SPO.RAB/#1,AMX/LA,ALU/A,SMX/ALU,SCK/LOAD"
SC_RC[]                  "SPO.R/LOAD.LC,SPO.RC/#1,BMX/LC,ALU/B,SMX/ALU,SCK/LOAD"
SC_R[].AND.R[]           "ALU/AND,AMX/LA,SPO.R/LOAD.LAB,SPO.RAB/#1,BMX/RMX,RMX/#2,SMX/ALU,SCK/LOAD"
SC_R[](EXP)              "SPO.R/LOAD.LAB,SPO.RAB/#1,AMX/LA,ALU/A,SMX/ALU.EXP,SCK/LOAD"
SC_RC[](EXP)             "SPO.R/LOAD.LC,SPO.RC/#1,BMX/LC,ALU/B,SMX/ALU.EXP,SCK/LOAD"
SC_SC+1                  "EALU/A+1,SMX/EALU,SCK/LOAD"
SC_SC.ANDNOT.FE          "EBMX/FE,EALU/ANDNOT,SMX/EALU,SCK/LOAD"
SC_SC.ANDNOT.K[]                   "KMX/#1,EBMX/KMX,EALU/ANDNOT,SMX/EALU,SCK/LOAD"
SC_SC+FE                 "EBMX/FE,EALU/A+B,SMX/EALU,SCK/LOAD"
SC_SC-FE                 "EBMX/FE,EALU/A-B,SMX/EALU,SCK/LOAD"
SC_SC+K[]                "KMX/#1,EBMX/KMX,EALU/A+B,SMX/EALU,SCK/LOAD"
SC_SC-K[]                "KMX/#1,EBMX/KMX,EALU/A-B,SMX/EALU,SCK/LOAD"
SC_SC.OR.K[]             "KMX/#1,EBMX/KMX,EALU/OR,SMX/EALU,SCK/LOAD"
SC_SC-SHF.VAL            "EBMX/SHF.VAL,EALU/A-B,SMX/EALU,SCK/LOAD"
SC_SHF.VAL               "EBMX/SHF.VAL,EALU/B,SMX/EALU,SCK/LOAD"
SC_STATE                 "EALU/A,MSC/LOAD.STATE,SMX/EALU,SCK/LOAD"
SC_STATE.ANDNOT.K[]           "EALU/ANDNOT,EBMX/KMX,MSC/LOAD.STATE,SMX/EALU,SCK/LOAD,KMX/#1"
SC&STATE_STATE-R[](EXP)  "LAE_R[#1],AMX/LA,EBMX/AMX.EXP,MSC/LOAD.STATE,EALU/A-B,SMX/EALU,SCK/LOAD"

;SD_... THRU VA_...

SD_NOT.SD                "SGN/NOT.SD"
SD_SS                    "SGN/SD.FROM.SS"
SS_CASE_D                "SGN/CLR.SD-SS"
SS_ALU15                 "SGN/LOAD.SS"
SS_SD                    "SGN/SS.FROM.SD"
SS_SS.XOR.ALU15&SD_ALU15           "SGN/SS.XOR.ALU"

STATE_0[A]               "AMX/RAMX.SXT,DT/LONG,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
STATE_AMX.EXP            "EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
STATE_D(EXP)             "RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
STATE_FE                 "EBMX/FE,EALU/B,MSC/LOAD.STATE"
STATE_K[]                "KMX/#1,EBMX/KMX,EALU/B,MSC/LOAD.STATE"
STATE_Q(EXP)             "RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
STATE_SC.XOR.KMX                   "MSC/LOAD.STATE,EALU/B,EBMX/KMX,KMX/SC"
STATE_STATE+1             "EALU/A+1,MSC/LOAD.STATE"
STATE_STATE.ANDNOT.FE          "EBMX/FE,EALU/ANDNOT,MSC/LOAD.STATE"
STATE_STATE.ANDNOT.K[]         "KMX/#1,EBMX/KMX,EALU/ANDNOT,MSC/LOAD.STATE"
```

```
STATE_STATE-FE   "EBNX/FE,EALU/A=B,MSC/LOAD,STATE"
STATE_STATE-FE   "EBNX/FE,EALU/A=B,MSC/LOAD,STATE"
STATE_STATE-K[]  "KMX/FF,EBNX/KMX,FALU/A=B,MSC/LOAD,STATE"
STATE_STATE-K[]  "KMX/FF,EBNX/KMX,FALU/A=B,MSC/LOAD,STATE"
STATE_STATE-OR.FE        "FALU/OR,EBNX/FE,MSC/LOAD,STATE"
STATE_STATE_OR.K[        "KMX/FF,EBMX/RMX,EALU/OR,MSC/LOAD,STATE"
;SPEC STATES
STATE_SKIPLONG           "STATE_K[.4]"
STATE_STATE.AK.SKIPLONG  "STATE_STATE.ANDKOT.K[.4]"
;DISP STATES
STATE_FIRST              "STATE_K[ZERO]"
STATE_PREDEC             "STATE_K[.6D]"
STATE_STATE.AK.STCC      "STATE_STATE.ANDKOT.K[.8F]"
STATE_STATE.AK.STC2      "STATE_STATE.ANDKOT.K[.7F]"
STATE_STATE.AK.DISTOBL   "STATE_STATE.ANDKOT.K[.6]"
STATE_STATE.AK.NOTPREDEC         "STATE_STATE.ANDNOT.K[.7F]"
STATE_STATE.AK.PREDEC2ERD        "STATE_STATE.ANDNOT.K[.D0]"
STATE_STATE_OR.ADVINP    "STATE_STATE.OR.K[.8]"
STATE_STATE.OR.JBS       "STATE_STATE.OR.K[.4]"
STATE_STATE.OR.DISTOBL   "STATE_STATE.OR.K[.6]"
STATE_STATE-OR.FILL      "STATE_STATE.OR.K[.7]"
STATE_STATE.OR.FLOAT     "STATE_STATE.OR.K[.60]"
STATE_STATE.OR.MOVS      "STATE_STATE.OR.K[.62]"
STATE_STATE.OR.PAT1      "STATE_STATE.OR.K[.1]"
STATE_STATE.OR.PAT2      "STATE_STATE.OR.K[.2]"
;MATCHING STATES
STATE_TRAEROBU   "STATE_K[.1]"
STATE_INNERSRC   "STATE_K[.3]"
STATE_OUTER      "STATE_K[ZERO]"

SWAPD            "DR/BYTE,SWAP"

VA_ALU           "VAX/LOAD"
VA_U             "RAMX/3,ANX/RAMX,ALU/A,VAX/LOAD"
VA_B.DAT[]+Q     "RAMX/3,ANX/RAMX,DAT,DT/81,BMX/RBUX,ALU/A+B,VAX/LEAD"
VA_B.ANDNOT.K[]  "RAMX/3,ANX/RAMX,BMX/RMX,MUX/81,ALU/ANDNOT,VAX/LOAD"
VA_D+4[]         "RAMX/3,ANX/RAMX,KMX/81,BMX/RMX,ALU/A+Q,VAX/LOAD"
```

```
VA_D+LE         "R+W+/D,+VX:+AMX,BMX:LC,ALU/A+B,VAK/LOAD"
VA_D+Q          "R+AM/D,+VX:+AMX,R+VX/Q,BMX/PSZA,ALU/A+B,VAK/LOAD"
VA_W[]          "WMX:01,B+X:+Ma,A,U/B,VAK:LOAD"
VA_-A           "AMX/LA,ALL/A,VAK:LOAD"
VA_LA,AND,LC    "AMX/LA,B+X/LC,A,U/AND,VAX/LOAC"
VA_LA,ENDNOT,KL!  "AMX/LA,  ,BMR/RMR,WMX:01,ALU/ANDNOT,VAK/LOAC"
VA_LA+D         "AMX/LA,BMW/D,B+X/REMA,ALU/A+B,VAK/LCAD"
VA_LA-D         "AMX/LA,BEU/D,BVX/RDMA,ALU/A-B,VAK/LOAD"
VA_.A+R[]       "AMX/LA,B+X/RMR,RMR/01,ALU/A+B,+AK/LOAD"
VA_LA-R[]       "AMX/LA,B+X/RMR,RMR/01,ALU/A-B,VAK/LOAD"
VA_LA+R[]       "AMX/LA,B+X/RMR,RMR/01,ALU/A+B,VAK/LOAD"
VA_LA+R[]+1     "AMX/LA,B+X/RMR,RMR/01,ALU/A+B+1,VAK/LOAD"
VA_LA+R[]-1     "AMX/LA,B+X/RMR,RMR/01,ALU/A+B-1,VAK/LOAD"
VA_LA+3         "AMX/LA,BMW/PC,ALU/A+B,VAK/LOAD"
VA_LA1Q         "AMX/LA,BMW/D,BMX/RSVD,ALL/A+B,VAX/LOAD"
VA_LA+Q         "VAK/LCAD,ALL/A-3,+MX/LA,BMR/RSVD,RBUX/Q,SHF/A,U"
VA_LRID.RX1     "B+X/LC,A,U/A-B,ANX/RSVD,BX1,D/BYTE,VAK/LOAD"
VA_SS           "B+X/PC,A,U/S,VAK/LCAD"
VA_Q            "RMR/Q,+MR/RSVD,ALL/A,+AK/LCAD"
VA_Q,ANDNOT,KL  "RMR/Q,+MR/RSVD,+XX/01,B+X/RSA,ALU/ANDNOT,VAK/LCAD"
VA_QID          "VAK/LOAC,ALL/RSVD,+MR/RMR,B+X/REMA,R+VX/Q,RBU4/D,SHF/ALU"
VA_Q+B[]        "RMR/Q,+MR/R+VX,+XX/01,B+X/RSA,ALU/A-B,+AK/LCAD"
VA_Q+R[]        "RMR/Q,+MR/R+VX,+XX/01,BMX/RMR,ALU/A-B,VAK/LCAD"
VA_Q-R[]        "RMR/Q,+MR/R+VX,+XX/01,BMX/LC,ALU/A+B,VAK/LOAD"
VA_Q1_C         "RMR/Q,+MR/R+VX,B+X/LC,ALU/A+3,VAK/LOAD"
VA_Q+1          "RMR/Q,+MR/R+VX,B+X/LC,ALU/A+1,VAK/LOAD"
VA_Q+1,B        "RMR/Q,+MR/R+VX,B+X/LC,ALU/A-B,VAK/LOAD"
VA_Q+LB,PC      "RMR/Q,+MR/R+VX,B+X/PC,3R,LB,ALU/A+B,VAK/LOAD"
VA_QIED         "RMR/Q,+MR/R+VX,B+X/PC,ALU/A-B,VAK/LOAD"
VA_R[]          "SSO,B/LCAD,L4E,SPL,RMR/01,+MR/LA,ALU/A,VAK/LOAD"
VA_RE[]         "SSO,B/LCAD,L4E,SPL,RMR/01,BMX/LC,ALU/S,VAK/LOAD"
VA_V+4          "PLV/V+4+4"

.TOC    "Non-trans-ar FLOAT JMS"

S.PDVK          "LOR.RSP1),QK/IC,OLR,IR,COND,PC_POIN,RUR/SPEB,V/6,PRSV"
BYTE            "DT/BYTE"
CACHE.INVALIDATE  "WRT/:WVALIDATE,VAK/WD"
CALL            "SUB/CALL"
CALL            "CALL,V/2!"
CALL/           "RLS/0 EC.../C.FORK"
CHK.RDD.ADDR    "VC/DIA.DUP.ADDR"
```

```
CHK.R1_LSS_0    "MSC/CHK,<LSS OFF"
CLR.L@SP        "CLR/<@SP>+"
CLR.L@R0        "CLR/<@R0>+"
CLRB_@R0+1      !@B/CLR 0,@
CLRB_@R0+2      !@B/CLR 0,@
CLRB_@R2+2      !@B/CLR 0,@
CLRB_@R2+3      !@B/CLR 0,@
CLRB_@R3+3      !@B/CLR 0,@
CLRB_@R4+3      !@B/CLR 0,@
CLRB_@R5+3      !@B/CLR 0,@
CLRB_@R6+3      !@B/CLR 0,@
CLRB_@R7+3      !@B/CLR 0,@
CLRB_@R8+3      !@B/CLR 0,@
CLRB_@R9+3      !@B/CLR 0,@
CLRB_@R10+3     !@B/CLR 0,@
CLRB_@R11+3     !@B/CLR 0,@
CLRB_@R12+3     !@B/CLR 0,@
CLRB_@R13+3     !@B/CLR 0,@
CLRB_@R14+3     !@B/CLR 0,@
CLRB_@R15+3     !@B/CLR 0,@
CLRB_@SP+3      !@B/CLR 0,@
CLRB_@PC+3      !@B/CLR 0,@
CLRB_@R0+SPEC   !@B/CLR 0,@R0+
CLRB_@R1+SPEC   !@B/CLR 0,@R1+
CLRB_@R2+SPEC   !@B/CLR 0,@R2+
CLRB_@R3+SPEC   !@B/CLR 0,@R3+
CLRB_@R4+SPEC   !@B/CLR 0,@R4+
CLRB_@R5+SPEC   !@B/CLR 0,@R5+
CLRB_@R6+SPEC   !@B/CLR 0,@R6+
CLRB_@R7+SPEC   !@B/CLR 0,@R7+
CLRB_@R8+SPEC   !@B/CLR 0,@R8+
CLRB_@R9+SPEC   !@B/CLR 0,@R9+
CLRB_@R10+SPEC  !@B/CLR 0,@R10+
CLRB_@R11+SPEC  !@B/CLR 0,@R11+
CLRB_@R12+SPEC  !@B/CLR 0,@R12+
CLRB_@R13+SPEC  !@B/CLR 0,@R13+
CLRB_@R14+SPEC  !@B/CLR 0,@R14+
CLRB_@R15+SPEC  !@B/CLR 0,@R15+
CLRB_@SP+SPEC   !@B/CLR 0,@SP+
CLRB_@PC+SPEC   !@B/CLR 0,@PC+
CLRB_@R0+DEST   !@B/CLR 0,@R0+
CLRB_@R1+DEST   !@B/CLR 0,@R1+
CLRB_@R2+DEST   !@B/CLR 0,@R2+
CLRB_@R3+DEST   !@B/CLR 0,@R3+
CLRB_@R4+DEST   !@B/CLR 0,@R4+
CLRB_@R5+DEST   !@B/CLR 0,@R5+
CLRB_@R6+DEST   !@B/CLR 0,@R6+
CLRB_@R7+DEST   !@B/CLR 0,@R7+
CLRB_@R8+DEST   !@B/CLR 0,@R8+
CLRB_@R9+DEST   !@B/CLR 0,@R9+
CLRB_@R10+DEST  !@B/CLR 0,@R10+
CLRB_@R11+DEST  !@B/CLR 0,@R11+
CLRB_@R12+DEST  !@B/CLR 0,@R12+
CLRB_@R13+DEST  !@B/CLR 0,@R13+
CLRB_@R14+DEST  !@B/CLR 0,@R14+
CLRB_@R15+DEST  !@B/CLR 0,@R15+
CLRB_@SP+DEST   !@B/CLR 0,@SP+
CLRB_@PC+DEST   !@B/CLR 0,@PC+
CLRB_@R0+R0     !@B/CLR 0,@R0+
CLRB_@R1+R1     !@B/CLR 0,@R1+
CLRB_@R2+R2     !@B/CLR 0,@R2+
CLRB_@R3+R3     !@B/CLR 0,@R3+
CLRB_@R4+R4     !@B/CLR 0,@R4+
CLRB_@R5+R5     !@B/CLR 0,@R5+
CLRB_@R6+R6     !@B/CLR 0,@R6+
CLRB_@R7+R7     !@B/CLR 0,@R7+
CLRB_@R8+R8     !@B/CLR 0,@R8+
CLRB_@R9+R9     !@B/CLR 0,@R9+
CLRB_@R10+R10   !@B/CLR 0,@R10+
CLRB_@R11+R11   !@B/CLR 0,@R11+
CLRB_@R12+R12   !@B/CLR 0,@R12+
CLRB_@R13+R13   !@B/CLR 0,@R13+
CLRB_@R14+R14   !@B/CLR 0,@R14+
CLRB_@R15+R15   !@B/CLR 0,@R15+
CLRB_@SP+R15    !@B/CLR 0,@SP+
CLRB_@PC+R15    !@B/CLR 0,@PC+
CLRB_@R0+R15    !@B/CLR 0,@R0+
CLRB_@R1+R15    !@B/CLR 0,@R1+
CLRB_@R2+R15    !@B/CLR 0,@R2+
CLRB_@R3+R15    !@B/CLR 0,@R3+
CLRB_@R4+R15    !@B/CLR 0,@R4+
CLRB_@R5+R15    !@B/CLR 0,@R5+
CLRB_@R6+R15    !@B/CLR 0,@R6+
CLRB_@R7+R15    !@B/CLR 0,@R7+
CLRB_@R8+R15    !@B/CLR 0,@R8+
CLRB_@R9+R15    !@B/CLR 0,@R9+
CLRB_@R10+R15   !@B/CLR 0,@R10+
CLRB_@R11+R15   !@B/CLR 0,@R11+
CLRB_@R12+R15   !@B/CLR 0,@R12+
CLRB_@R13+R15   !@B/CLR 0,@R13+
CLRB_@R14+R15   !@B/CLR 0,@R14+
CLRB_@R15+R15   !@B/CLR 0,@R15+
CLRB_@SP+R15    !@B/CLR 0,@SP+
CLRB_@PC+R15    !@B/CLR 0,@PC+
CLRB_@R0+R0     !@B/CLR 0,@R0+
CLRB_@R1+R1     !@B/CLR 0,@R1+
CLRB_@R2+R2     !@B/CLR 0,@R2+
CLRB_@R3+R3     !@B/CLR 0,@R3+
CLRB_@R4+R4     !@B/CLR 0,@R4+
CLRB_@R5+R5     !@B/CLR 0,@R5+
CLRB_@R6+R6     !@B/CLR 0,@R6+
CLRB_@R7+R7     !@B/CLR 0,@R7+
CLRB_@R8+R8     !@B/CLR 0,@R8+
CLRB_@R9+R9     !@B/CLR 0,@R9+
CLRB_@R10+R10   !@B/CLR 0,@R10+
CLRB_@R11+R11   !@B/CLR 0,@R11+
CLRB_@R12+R12   !@B/CLR 0,@R12+
CLRB_@R13+R13   !@B/CLR 0,@R13+
CLRB_@R14+R14   !@B/CLR 0,@R14+
CLRB_@R15+R15   !@B/CLR 0,@R15+
CLRB_@SP+R15    !@B/CLR 0,@SP+
CLRB_@PC+R15    !@B/CLR 0,@PC+
CLRB_@R0+R15    !@B/CLR 0,@R0+
CLRB_@R1+R15    !@B/CLR 0,@R1+
CLRB_@R2+R15    !@B/CLR 0,@R2+
CLRB_@R3+R15    !@B/CLR 0,@R3+
CLRB_@R4+R15    !@B/CLR 0,@R4+
CLRB_@R5+R15    !@B/CLR 0,@R5+
CLRB_@R6+R15    !@B/CLR 0,@R6+
CLRB_@R7+R15    !@B/CLR 0,@R7+
CLRB_@R8+R15    !@B/CLR 0,@R8+
CLRB_@R9+R15    !@B/CLR 0,@R9+
CLRB_@R10+R15   !@B/CLR 0,@R10+
CLRB_@R11+R15   !@B/CLR 0,@R11+
CLRB_@R12+R15   !@B/CLR 0,@R12+
CLRB_@R13+R15   !@B/CLR 0,@R13+
CLRB_@R14+R15   !@B/CLR 0,@R14+
CLRB_@R15+R15   !@B/CLR 0,@R15+
CLRB_@SP+R15    !@B/CLR 0,@SP+
CLRB_@PC+R15    !@B/CLR 0,@PC+
```

```
RETURN0         .SJM/RET.J/60
RETURN1         .SJM/RET.J/61
RETURN10        .SJB/RET.J/10
RETURN0C        .SJB/RET.J/0C
RETURN10F       .SJB/RET.J/10F
SET.OCL1INST1   .CCK/INST.DSP.DT/INST.DEP
SET.OCL1LONG1   .CCK/INST.DSP.DT/LONG
SET.DEC(R0A)    .CCK/DEC
SET.FPD         .WSC/SET.FPD
SET.V.ARC.Z     .CCK/TST.Z
SET.VBIT.ERR    .WSC/SET.VBIT.ERR
SET.N&Z         .CCK/N&Z_PLI
SET.PSL.CI4MKI  .CCM/CC_MASK
SET.V           .CCM/SET.V
START.IB        .JEC/START
STOP.IB         .JEC/STOP
TEST.IB.PCHK    .VCT/TEST.PCHK,VAK/NOP
TEST.IB.MCHK    .VCT/TEST.MCHK,VAK/NOP
TRAP.ACC[]      .ACF/TRAP,ACM/8
NOOP            .DT/NOOP
WRITE.RFST      .VAB_R(SP+1),OK/13,CLR:18,COND_PC_PCH+1,SUB/SREG,V/XRD

.TOC    Branch Enable Macro Definitions

ACCEL?          .BEN/ACCEL
ACC.SYNC?       .BEN/ACCEL
AC.LOST         .BEN/IN.ESRJPT
A.10MEDT        .BEN/IS.TEST
ALU?            .BEN/ALU
ALU.NT          .BEN/ALU
ALU=0?          .BEN/ALU=0
BCDSGN?         .BEN/DECIMAL
C31?            .BEN/C31
CONSOLE.MODE?   .BEN/CONSOLE.MODE
D0?             .BEN/D0=0
D1??            .BEN/MU
D2?             .BEN/D0=0
D2-0?           .BEN/D0=0
D3?             .BEN/D0=0
D3-0?           .BEN/D0=0
D3?             .BEN/STORE
```

```
DATA.TYPE?      "DEN/DATA.TYPE"
D.EQ?           "DEN/D.NYTES"       ;.04/0E"
D.GE?           "BEN/D.BYTES"       ;.04/0C"
D.GT?           "BEN/D.BYTES"       ;.04/0B"
DE.?            "BEN/DATA.TYPE"
D.LE.TES?       "BEN/D.BYTES"       ;.03/5" ;PREFERED FORM
D.NE.0?         "BEN/D.ONE"
EALU?           "BEN/CALL"
CALU.N?         "BEN/EALL"          ;.04/07"
EALU.Z?         "BEN/EALL"          ;.04/03"
END.D-1?        "BEN/ERO.OMI"
FPOT            "BEN/ASI.REF"       ;.04/01"
FP.TEST?        "BEN/IS.IESI"
INT?            "BEN/INTERRUPT"
INTERRUPT.REQ?  "BEN/INTERRUPT"     ;.03/5"
ISC1            "BEN/ALU"           ;.04/0D"
ISC.C1?         "BEN/ALU"
ISI             "BEN/IRD.1"         ;.03/6"
ISI-1?          "BEN/IRD.0"
LAST.REF?       "BEN/.AST.REF"
MODE.SELECTED?  "BEN/REG"           ;.03/9"
NU.1            "BEN/VUL"
NEST.RET?       "BEN/.AST.REF"      ;.04/05"
PC.MODES?       "BEN/PC.MODES"
PSL.C?          "BEN/PSL.CC"        ;.04/0E"
PSL.CC?         "BEN/PSL.CC"
PSL.MODE?       "BEN/PSL.MODE"
PSL.N?          "BEN/PSL.CC"        ;.04/7"
PSL.V?          "BEN/PSL.CC"        ;.04/0D"
PSL.Z?          "BEN/PSL.CC"        ;.04/0C"
PTE.VALID       "BEN/TB.TEST"       ;.05/0F"
QUAD?           "BEN/DATA.TYPE"
R.EQ?           "BEN/SIGNS"         ;.03/3"
R.EQ.ENTY?      "BEN/ALU-0"         ;.04/7"
RDVT            "BEN/ROM"
SC1             "BEN/SC"
SC.GT.32        "BEN/SC"
SC.NE.00        "BEN/WIL"           ;.03/3"
SIGNS?          "BEN/SIGNS"
SRC.PC?         "BEN/SRC.PC"        ;COMP MODE, BEN OK SRC R = PC
SSI             "BEN/EALL"          ;.04/0E"
STATE0?         "BEN/STATE0-0"      ;.04/05"
STATE1?         "BEN/STATE0-0"      ;.04/03"
```

```
;FIELDS ARRANGED FROM HI TO LO ORDER BITS

.RTOL
.HEXADECIMAL

NAD/=0,8.39,=                   ;NEXT ADDRESS DEFAULT IS THE FOLLOWING
                                ;MICRO WORD
                                ;
BEN/=0.8.30,U                   ;BRANCH TABLE

        NEP=0                   ;DEFAULT
        BEN1=1                  ;BEL TABLE FOR EXPLANATION
        BEN2=2                  ;
        BEN3=3
        BEN4=4                  ;
        BEN5=5                  ;
        BEN6=6                  ;
        BEN7=7

AMXC/=2,2,54.D                  ;BALU A INPUT


        LA=0                    ;SEL LA TO BALU
        LB=1                    ;SEL LB TO BALU
        PR=2                    ;SEL PR TO BALU
        COND=3                  ;ADD. CONDITIONAL

BMXC/=0,2.32,D                  ;BALU B INPUT


        MD=0                    ;MD/ NORMALIZATION CONSTANT
        A=1                     ;A REGISTER
        ABC=2                   ;RESTORE DIVIDE (C(*8) NORMATION
        IR=3                    ;SEL IR TO BALU


BALUC/=0,2,30.D                 ;BALU CONTROLS
```

```
        A=B             ;PASS A/B/A
        A-B=1           ;DIVK MINUS BMA
        A+B=2           ;ADD PLUS /MA
        ALU=3           ;EALU=OFF

-PSYNC/=0,1,29,0        ;SYNC TO CPU

        XOP=0
        P=S=1           ;ASSERT /PS=NC


MCTL/=0,1,30,1


        XOP=0
        CNT=1           ; START A MULTIPLY

EXC/=0,4,24,0           ;EXPONENT PROCESSOR CONTROLS

        XOP=0
        LDX=1           ; POLY ARG EXP REG GETS EALU
        LDP=2           ; PRODUCT EXP REG GETS EALU
        PR.XR=3         ; LOAD PR & XR
        LDE=4           ; LB GETS BUS B
        XR.LB=5         ; LB GETS BUS, XR GETS EALU
        PR.LB=6         ; LB_BUS, PR_EALU
        PR.XR.LB=7      ; LB_BUS, PR&XR_EALU
        LDA=8           ; LA_BUS A
        XR.LA=9         ; LA_BUS, XR_EALU
        PR.LA=0A        ; LA_BUS, PR_EALU
        PR.XR.LA=0B     ; LA_BUS, PR&XR_EALU
        LDAS=0C         ; LA_BUS A, LS_BUS 4
        XR.A4=0D        ; LA_BUS A, LB_BUS B, XR_EALU
        PR.AB=0E        ; LA_BUS A, LB_BUS B, PR_EALU
        LD.ALL=0F       ;
```

```
WAIT/=0,1,BS,D

        NW/=0                   ;
        WAIT=1                  ;ENABLE STALL

MSC/=0,3,20,D                   ;MISCELLANEOUS CONTROLS

        NOP=0                   ;
        =ALU=1                  ;POLY ADD FOR SIGN PROCESSOR
        MS=2                    ;SET SIGN BIT FOR RESULT OPND
        =R.D=3                  ;CLEAR REMAINDER REGISTER
        CC=4                    ;ERROR CONDITIONS
        LDX=5                   ;LOAD SIGN OF 0 FOR POLY
        IBD=6                   ;INSTRUCTION BRANCH 0
        IB1=7                   ;INSTRUCTION BRANCH 1

NRL/=0,2,18,D                   ;NORMALIZER REGISTER

        NOP=0                   ;
        S_=1                    ;SHIFT LEFT  (NORMALIZE)
        LD=3                    ;LOAD BUS A, BUS B

SCR/=0,2,16,D                   ;FPA SCRATCH PAD CONTROLS

        CPU=0                   ;REGISTER ADDRESS FROM CPU
        DP6.R=1                 ;SP2[1],PR3=1
        R.R=2                   ;SP1,SP2
        DP=3                    ;PRN[1], PRN
```

```
BSC/=8.4.12,0                   ; DATA SOURCE FOR BUS A AND BUS B

        INTM=0                  ;INTEGER PRODUCT MI TO BUS A
        NL=1                    ;NORMAL TO BUS A  BUS A TO BUS B
        NM=2                    ;NORMH & FAT TO BUS A, A TO B
        PQ=3                    ;PROD/QUOTH TO BUS A, P/QL TO BUS B
        INTL=4                  ;INTEGER PRODUCT TO BUS A
        ID=5                    ;ID BUS TO BUS A, B
        LR=6                    ;IBUS DATA REGISTER TO BUS
        ID.RR=7A                ;ID TO BUS A, R1 TO BUS B
        R=8B                    ;RA TO BUS A, R4 TO BUS B
        FAL.X=9C                ;HARDWARE DETERMINED
        FAL.LH=AD               ;FALLL TO BUS A, FATCH TO BUS B
        FAL.=LFCE               ;FALLH TO BUS A, FALLL TO BUS B

FADC/=0C,4,3,0                  ;FRACTION PROCESSOR CONTROLS

        AR=0                    ;LOAD AR1, AR0
        BR1=1                   ;LOAD BR1
        AR1=2                   ;LOAD AR1
        R1=3                    ;LOAD AR1, BR1
        SR=4                    ;LOAD SR1,BR0
        BR0=5                   ;LOAD BR0
        AR0=6                   ;LOAD AR0
        RC=7                    ;LOAD BR0, AR0
        LD=8                    ;LOAD AR, BR
        A.B=9A                  ;HARDWARE DETERMINED ADD/SUB
        A=0C                    ;OUTPUT AR
        B=0D                    ;OUTPUT BR

SGNC/=0.3.5.2                   ;SIGN LATCH CONTROLS

        NOP=0                   ;SIGN LATCHES UNCHANGED
        LDSA=1                  ;SIGN BIT TO SA
        A.SA=2                  ;IF SUB, SA<= NOT SA ; ELSE SA <= SA
        A.P=3                   ;RESULT SIGN TO SA
```

FPA CONTROL ROM FIELD DEFINITIONS

```
        LDSB=4              ;BUS B(15) TO SB
        LDS=5               ;BUS A(15) TO SA  BUS B(15) TO SB
        A,B=6               ;SB TO SA
        A,XOR,X=7           ;SA XOR X TO SA

LRR/=0,1,4,D                ;REMAINDER REGISTER CONTROLS

        NOP=0               ;
        LD,RR=1             ;LOAD REMAINDER REGISTER

OPLD/=0,4,0,D               ;MULTIPLIER OPERAND CONTROLS

        NOP=0               ;
        MC=1                ;LOAD MULTIPLICAND
        MC1=2               ;LOAD UPPER HALF OF MULTIPLICAND
        MC,FD=4             ;LOAD DP  VM HALVES
        MC0=5               ;LOAD LOWER HALF OF MULTIPLICAND
        LDR,C=8             ;LOAD ALL FOR R-R
        MCI,MPI=0A          ;LOAD M'CAND INT SEA&MULTIPLIER HIGH FRACTION
        MCI=0B              ;LOAD MULTIPLICAND INTEGER
        MP=0C               ;LOAD MULTIPLIER
        MP0=0D              ;LOAD LOWER HALF OF MULTIPLIER
        MP1=0E              ;LOAD MULTIPLIER, HIGH FRACTION
        INIT=0F             ;CONTROL INITIALIZATION

        ****************************************
        *                                      *
        *              BEN TABLE               *
        *                                      *
        ****************************************

;BEN    UADRS<2>        UADRS<1>        UADRS<0>
;---    --------        --------        --------
;0      0               0               0       (NOP)
```

```
;1      F_DAT H         IR9R) L         IR9R) L (OPCODE SYNCh)
;
;2      SAR H           SAR H           SAR H   (NORMALIZATION SHIFT W/THIN RANGE)
;
;3      ABV H           D=0 H           A=C H   (ZEROES AND RESERVED OPERANDS)
;
;4      POLY DONE L     CPSYNC H        FLOAT H (SYNCHRONIZATION WITH CPU)
;
;5      (A OR B)=0 H    SubExpO H       SubExpDiff H        (EXP. DIFF.)
;
;6      0               0               MUL/DIV DONE H
;
;7      0               [PreO+PrO&>] L  PrVE> H (OVER/UNDERFLOW IN POLY)
;
        *TRANSFER MACRO DEFINITIONS*

AR_PROD                 *BSC/PQ,EAQC/AR*
BFORK                   *MSC/LRI,NAJ/100.WAIT/AAI*
BJS_C                   *BSC/NL,EA.JC/RDFF,AMXC/PR*
CPU_ANSH                *EMXC/NR,BSC/NH,AMXC/PR.EALUC/A*
CPU_ANSL                *EMXC/NR,BSC/NL,AMXC/PR,EALUC/A*
EALU_PR                 *ANXC/PR,EALUC/A*
EALU_PR-XR              *ANXC/PP,EMXC/XR,EALUC/A-B*
ERCH                    *MSC/CC*
FPA.IRD                 *SCR/R.R.IMXC/A*,MSC/IRD.ESC/R.SGNC/LDS.OPLD/LDA.R,EAC/LDA9*
FPSYNC                  *!PSYNC/FPS*
IRD                     *MSC/OAD,/A.JC/S,FAUL/LD.SGNC/A.RES,BSC/NA,OPLD/INIT,MSC/RR.D*
LAAPR_LB                *BSC/NH,AMXC/LB.EALUC/A,EAC/PR.LA*
LAAPR_PR-M              *AMXC/PR.EALUC/A-B,EMXC/MSO,BSC/NH,EAC/PR.LA*
LASS4_0                 *EALUC/K-1.PRESC.NR,BSC/NH.EAC/LDA,SGNC/A.RES,AMXC/PR*
LA_C                    *EALUC/A3FF.ENXC.NR,MSC/NH.EAC/LDA,AMXC/PR*
LA_LB                   *AMXC/LB,EALUC/A,BSC/NH.EAC/LDA*
LA_PR                   *AMXC/PR,EALUC/A,EMXC/LB,BSC/NH.EAC/LDA*
LA_PR-M                 *AMXC/PR,EALUC/A-B,EMXC/MSO,BSC/NH,EAC/LDA*
LA_PR-N                 *AMXC/PR,EALUC/A-B,EMXC/MSO,BSC/NH,EAC/LDA*
LD_D                    *EALUC/K3FF,ENXC.NR,BSC/NL,EAC/LDB.SGNC/LDSB*
LB_PR                   *AMXC/PR,EALUC/A,EMXC/LB,BSC/NH.EAC/LDB*
```

```
;B_PHI0
;DAC_RD
;DAC_WR
;DAC_CSEL
;DAC_CNT
;DAC_ZSE
;DAC_NSHFL
;DAC_PSHFH
;CCLK
;MC_SR
;HBI
;ND
;CTRL_FD
;NC_SR
;NB_ADD
;NB_DOUT
;CLP_MOD
;NB_C
;NB_CONV
;NB_K
;NB_A
;NB_A-1
;NB_ALU
;NB_A-MSCN
;NB_B
```

TRANSFER MACRO DEFINITIONS

```
FR_LE+K                 *AMXC/LE,EALUC/A-E,EMXC/XEQ,EAC/LDPR*
FR_LT+K                 *AMXC/LE,EALUC/A-E,EMXC/XGQ,EAC/LDPR*
FR_LE-AR                *AMXC/LE,EMXC/XR,EALUC/A-B,EAC/LDPR*
FR_PR+K                 *AMXC/PR,EMXC*K,EALUC/A+B,EAC/LDPR*
FR_PR+NROW              *AMXC/PR,EMXC*NROW,EALUC/A+B,EAC/LDPR*
FR_PR+R                 *AMXC/PR,EMXC/*R,EALUC/A+B,EAC/LDPR*
FR_PR-R                 *AMXC/PR,EMXC/*R,EALUC/A-B,EAC/LDPR*
FR_UNDR                 *EAC/LDPR,EALUC/A-1*
FR_XY+LB                *EAC/LDPR,EMAC/*LB,EMXC/XR,EALUC/A+B*
FR_XR                   *LDR/LD XR*
FRX                     *MSC/MD*
SA_EA,XOR,SUB           *SGNC/A,SUB*
SA_EA,XOR,SX            *SGNC/A,EALUC/A,X*
SA_EA                   *SGNC/A,B*
SA_SR                   *SGNC/A,MSC*S*
MX_SA                   *MSC/LDSA*
WAIT                    *WAIT/WAIT*
XR_LA                   *AMXC/LA,EALUC/A,EAC/LDXR*
```

BRANCH MACRO DEFINITIONS*

```
JMP,UNREL,D?            *BEN/8*
OPSYNC?                 *BEN/4*
OVFLOW?                 *BEN/6*
ERROR?                  *BEN/7*
DEFJOB?                 *BEN/5*
FLOAT?                  *BEN/1*
MUL,DONE?               *BEN/E*
DIVIDE?                 *BEN/F*
SWR?                    *BEN/2*
XTRJMP?                 *BEN/C*
POLY,DONE?              *BEN/A*
```

SECTION 6
TROUBLESHOOTING TOOLS/DIAGNOSTICS

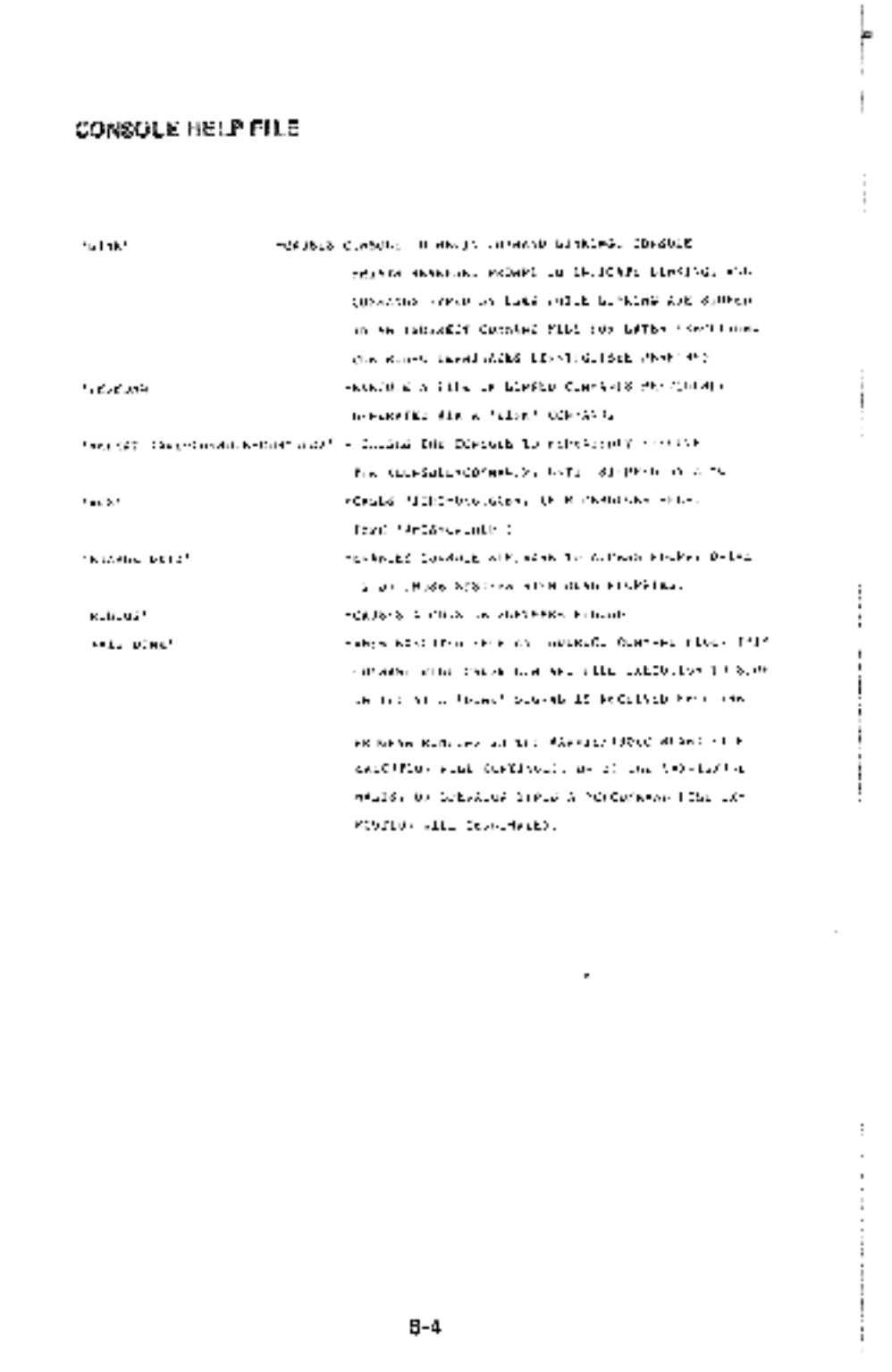
CONSOLE HELP FILE

```
/HX-11/TBV CONSOLE HELP FILE     V1.1, 4-JUN-79 11:2
GENERAL:        Following is a summary of the commands which may be issued
                after the prompt is given.
                Arguments in square brackets are optional.  All numbers are
                assumed to be octal unless followed by a decimal point (.).
                'xx,yy,zz,...,ww' indicates that multiple arguments may be
                'cmd'   (Perform command)
                'n'     (Load n words)
                '-'     (Examine next location 'n' times)
                '^'     (Examine previous 'n' locations)
                '/'     (Open location 'n' and type contents)

                Characters in a word or line may be deleted by typing
                DEL or RUBOUT.  If it is desired to delete a character
                from a line, type CTRL/U.

        ALL COMMANDS ARE TERMINATED BY CARRIAGE RETURN
'BREAK': Suspends          -ISPLAYS CURRENT CONTENTS OF REGISTERS
'DISPLAY [SUSPENDED]'      -ALLOWS USER TO RESUME
                DEPOSIT [n] [,n] [n] DEPOSITS [n] INTO [n]
                EXAMINE [n] [,n] [n] EXAMINES [n] [,n] [n]
                '/'     OPEN PREVIOUS LOCATION
                '//'    OPEN NEXT LOCATION
                '/I'    OPEN INSTRUCTION REGISTER
                '/S'    OPEN SUBROUTINE RETURN STACK
                '/R'    OPEN 'R' REGISTER
                '/LD'   OPEN LOAD REGISTER
        EXAMPLE: TO EXAMINE CONTENTS OF LOCATION 1000, USE 0:1000
                DEFAULT CONTROL STRING FOR '//' LOOKS

'CONTINUE'              -TARTS EXECUTION FROM REGISTER CONTENTS.
                         IF CURR. INSTR. IS BREAK, EXECUTION BEGINS AT CURRENT
'START [ADDRESS]'       -INITIALIZES THE CPU, CLEARS A CONSOLE
                         TO THE PC, STARTS A CONTINUE FROM THE PC.
'CONTINUE'              -STARTS A CONTINUE TO THE CPU.
'HALT'                  -HALTS THE CPU.
'RUN'                   -RUNS THE CPU FROM INITIAL DEVICE
'INITIALIZE'            -INITIALIZES THE CPU
```

ÇOMSOLE HELP FILE

```
'CMDline'               -DISPLAYS CURRENT CONSOLE AND CPU STATUS
'CMD <> BREAKPOINT'     -DISPLAYS REGIONS IN MICROCODE AND CONSOLE
'CLOCK'                 -RUNS MICRO-DIAGNOSTICS
'DEBUG/ICE'                     -CALLS MICRO-DIAGNOSTIC 'CMDLINE,BREAK' [ MP-CONS
'UNJAM'                 -UNJAMS THE BUS
'SET BOOT BJB'          -MAKES FLOPPY DISK 0 BOOTABLE
'SET BOOT HARD'                 -MAKES WINCHESTER DISK STATE CURRENT BOOT
'SET STEP INSTRUCTION'  -ALLOWS ONE CONSOLE INSTRUCTION TO RUN
'CLEAR ALL'                     -CLEARS ALL BREAKPOINT STEP MODE
'CLOCK <FREQUENCY>'             -CLOCKED SCOPE CYCLE AND DATA ZERO U: STAT
                                 UNDEFINED IN CASE 'SET BOOT' CURRENT
'CONFIG <MODEMSG>'              -PROB A WORD CLEAR TO CONFIG=0,1:4  0 CONFIG
                                 IS A BASE ADDRESS FOR MODEMSG BOARD (IF PRESENT)
'SLP DOWN'                      -SETS 'STOP ON WATCHDOG' BATTERY BACKUP
'CLEAR DOWN                     -CLEARS 'STOP ON WATCHDOG' BATTERY BACKUP
'SET CLOCK SLOW'                -SET CPU CLOCK RATE TO 4.
'SET CLOCK FAST'                -SET CPU CLOCK RATE TO 8MHZ
'SET CLOCK 8086'                -SET CPU CLOCK RATE TO 10MHZ
'SET BOOTFILE'[1<= #<=8>        -SETS CONFIGURATION FILE FOR BOOTING TO BE SPECIFIED
                                 BOOTFILE. IF NO FILE IS SPECIFIED THE DEFAULT
                                 IS CONFIG.SYS IN THE ROOT DIRECTORY.
'<> [ DRIVE:] <1-4 LETTERS>[.<1-3 LETTERS>]'    -SETS CONSOLE DEFAULT
       DRIVE TO BE <1-4 LETTERS> LONG       DRIVE:=A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
                                 <NAME>.<EXT>,<NAME>.EXE,<NAME>.COM
'SET TERMINAL FILE[<CR>FNAME]'  -SETS LOG FILE NAME FOR ALL CONSOLE OUTPUT
                                 (IF NO  NAME GIVEN THEN CLOSES LOG FILE)
'SET TERMINAL PAUSE[ON]' -SETS CONSOLE  BAUDRATE  (MUST MATCH BAUD RATE OF TERM
'?'[C <CR> ]<--]                -PRINTS INFORMATION TO ALL 'DISPLAY' COMMANDS
                                -DISPLAYS NEXT SCREEN IN A 'DISPLAY' PROCESS
'<ALT>'                 -PRINTS LOCK FILE
'<FILE=FNAME>'                  -PROCESS AN INDIRECT COMMAND FILE
'LOAD <FILENAME>'               -LOAD FILE TO RAM MEMORY.
'LOAD/GO <FILENAME>'            -LOAD FILE TO RAM AND START IT
       NOTE:   THE '<>[#]<>[<FNAME>]' SYNTAX FOR SETTING BOOT FILE IS ALSO
               FOR X.25 SETUP SYNTAX FOR A CONSOLE. IF NECESSARY BOTH SYNTAX
               MAY BE USED AT A TIME.
```

CONSOLE HELP FILE



CONSOLE ABBREVIATION RULES

```
[illegible] CONSOLE ABBREVIATION Rules   Rev 0.0/7.77
THIS FILE SAVED FOR SYSTEMS USING CONSOLE STATUS FILE MODE 0.
RECOGNIZED AS THE C MODEL CONSOLE TERMINAL.

      C STATUS                  CONSOLE COMMAND ABBREVIATION RECOGNIZED
'CONSOLE CONFIGURATION'         'C CONFIGURE'
'CONSOLE CONFIGURE <OPTION>'    'C CONFIGURE <OPTION>'
'QUERY CONSOLE'                 'Q CONFIGURE'
'CONTINUE'                      'C'
'CALL'              'A'
'HELP'              'H'
'QUIT'              'Q'
'INITIALIZE'                    'I'
'ONLINE'            'ON'
'ON-LINE VERSION'   'ON V'
'TEST'              'T'
'VERIFY'                        'V'
'DISK PARAMETER/CONFIGURATION'  'SE D:PARAMETER'
'DISK SCRP MAP'     'SE S M'
'DISK SCRP SCALE'               'SE S S'
'DISK SCRP VERIFY'  'SE S V'
'CLEAR STAT'                    'CL S'
'SAVE CONFIG'                   'W CONFIG'
'CONFIG CONFIGURE'              'W CONFIGURE'
'SET DATE'                      'SE DA'
'CLEAR SCREEN'                  'CL S'
'SET CLOCK FREQ'                'SE C F'
'SET CLOCK SYNC'                'SE C S'
'SET CLOCK NORMAL'              'SE C N'
'DEFINE [illegible]'            'DE [illegible]'
```

CONSOLE ABBREVIATION RULES

'LOAD <FILENAME>'	'L <FILENAME>'
'LINK' 'LI'	
'RUN'	'R'
'RUN <FILENAME>'	'R <FILENAME>'
'REG'	'R'
'ERASE DEL'	'ER DEL'
'REBOOT'	'REB'
'SET TERMINAL FILL(<NUMBER>)'	'SE T F(<NUMBER>)'
'SET TERMINAL PAUSE(N)' 'SE T P(N)'	
'SET JOB(ALL <OPTION>[,N])'	'SE J <OPTION>[,N]'
'HELP <CMD>'	'HE <CMD>'

QUALIFIERS		CHARACTER ABBREVIATION RECOGNIZED
/DEVICE		/D
/RECORD		/R
/COMMAND		/C
/LOAD		/L
/OCTAL	/O	
/HEX		/H
/PRIVILEGED		/P
/[NO]WAIT		/W
/[NO]MESSAGE		/M
/[NO]ABORT		/A
/[NO]LOG		/LO
/[NO]ERROR		/E
/[NO]CONTROL		/CO
/[NO]ATTACH		/AT
/[NO]PRIVILEGE		/PR
/[NO]CLEAR		/CL
/[NO]SYSTEM		/SY
/[NO]SHARE		/SH
/[NO]ACCESS		/AC
/[NO]ALLOCATE		/AL
/[NO]DEALLOCATE		/DE
/[NO]ASSIGN		/AS
/[NO]DEASSIGN		/DEA
/[NO]MOUNT		/MO
/[NO]DISMOUNT		/DI
/[NO]INITIALIZE		/IN
/[NO]SPOOL		/SP
/[NO]SHUTDOWN		/SHU
/[NO]CREATE		/CR
/[NO]DELETE		/DEL
/[NO]RENAME		/REN
/[NO]PROTECTION		/PRO
/[NO]MODIFY		/MOD
/[NO]ACCOUNT		/ACC
/[NO]BACKUP		/BA
/[NO]INSTALL		/INS
/[NO]REMOVE		/REM
/[NO]CHECKPOINT		/CH
/[NO]RESTORE		/RES
/[NO]SAVE		/SA
/[NO]SETPRIVILEGES		/SE
/[NO]INSTALLED		/INST

```
BREAK TABLE          =ESTABLISH SEQUENCES TO TRANSMIT TO/FROM REMOTE
                      ULTIMATE LOCAL AND REMOTE TERMINALS. THE
                      SEQUENCE IS ENTERED AS A HEXADECIMAL STRING
                      WHERE EACH TWO CHARACTERS REPRESENT ONE BYTE.

CONTROL CHAR         =CHOOSE CHARACTERS TO BE USED AS YOUR ESCAPE TO
                      ACCESS MODE IN THE REMOTE TERMINAL.

CHANGE LOCAL CAPS    =CHOOSE YES  IF YOU WANT YOUR LOCAL TERMINAL
                       TO OUTPUT BOTH UPPER AND LOWER CASE TO THE

CHANGE REMOTE CHARACTERISTICS=CHOOSE THIS WHEN TRANSMITTING TO CONTROL THE REMOTE
                      TERMINAL SUCH AS IF THE REMOTE TERMINAL IS
                      NEEDED BE A CONTROL-P CHAR TO SEND  CHARACTERS

CHANGE CARRIER TIMEOUT=CHOOSE THE NUMBER OF SECONDS YOU WANT TO WAIT
                      FOR A LOSS OF CARRIER BY MODEM BEFORE DISCONNECTING.

DISPLAY MODE         =SPECIFIES SCREEN OF CHARACTERS  WHICH IS VIDEO TERM.

DISPLAY LOCAL CAPS   =DISPLAYS A YES IF YOU ARE  YOUR TERMINAL CAPS OR
                      OUTPUT TO REMOTE TERMINAL.

DISPLAYS CARRIER BAUD=DISPLAYS THE BAUD AT WHICH PROBLEMS OF CARRIER
                      LOSS  REMOTE PORT LOSS OF CARRIER DETECTED.
```

```
                                    RE**E   MAY 1977
ICO STOP POINTING, TYPE 'G'
?
 UNRECOGNIZED COMMAND/ERROR DESCRIPTION OF CARRIAGE RETURN!

'E^R <ADDRESS>'                     -DISPLAY PHYSICAL MEMORY
'E^I <ADDRESS>'                     -DISPLAY I/O PORT CONTENTS

'E <ADDRESS>'                       -DISPLAY 4CH B CATION, DISPLAY CELL CONTENTS
'M <SOURCE> <F]-]:M#4#>],<M SOURCE>-M>,,,<-INCREMENTS>
                            ASSIGN 4CH LOCATION, DISPLAY OLD PCONTENTS
                            THE FIELDS SPECIFIED.
        <BIAS <FIELDMP4CS> =    ACT.PCP,ACC.ALU.SSP,D*X.CCS.CSI,IN.DI.BUS
                                SUP.PSP.PS.IC.IEN.JUP.RAR.ACT,MEC.JUS.SS
                                JNS.SCP.BSM.SMF.EI.PNS.B/D.CCU,PM

'E BA <ADDRESS>'                    -DISPLAY RM RA REGISTER
'M P, <SOURCE>'                     -DISPLAY RM RC REGISTER

'M SA,<SOURCE>-<SIZE>               -DISPLAY CMA OF CMA SCRATCH/SMALL MEMORY
                                    BLOCKS
        <BIAS <SYSMON.C-SYSMON> = PA,IR,IAC,DA,BA,AC..,MA,DI.SR,AC

'E/P <ADDRESS> <DATA>' -DISPLAY DATA AT PHYSICAL ADDRESS
'E/I <ADDRESS> <WORD>'              -DISPLAY IO ID & REGISTER

'E <ADDRESS> <FIELD,MP#E-#> <DATA-1>,<FIELD,DATA-2> <WORD-2>,.....
                                    -DISPLAY IO ACC LOCATION, (FIELD, SOURCE-1)
                                    AND (FIELD,SOURCE-2), THEN INCREMENT
                                    ADDRESS FOR EACH DATA
                                    WORD ENTERED.
        <BIAS IOB '/a' ,/DI:DIA MAY BE TAKE AT CHANGE DIA DISPLAY AD
                FIELDS  0 TO C.MAP:0).
```

MICRODEBUGGER HELP FILE

```
'D RA <ADDRESS> <DATA>'          -DEPOSIT <DATA> TO RM RA REGISTER
'D BC <ADDRESS> <DATA>'          -DEPOSIT <DATA> TO AM BC REGISTER

'D <COMMAND-NAME> <DATA>'        -DEPOSIT <DATA> TO USE OF THE MEMORY-MAPPED
                                  BANKED REGISTER/BLOCK (JAT ENTRY).
      NOTE: DEPOSITS TO THE SUCH SCRATCHPAD ARE NOT AUTHORIZED.

'CONTINUE'                       -RESUME MICRO-INSTRUCTION EXECUTION AT
                                  SPECIFIED MC CONTENTS OF MICRO-PC(JPC)

'START <ADDRESS>'                -START MICRO-INTERPRETER AT <ADDRESS>.

'HALT'                  -HALT THE MICRO-PROCESSOR

'RST SUPH'                       -SET THE 'STOP ON MICRO-PAUSE' LATCH
'CLEAR SUPH'                     -CLEAR THE 'STOP ON MICRO-PAUSE' LATCH

'SET STEP'                       -ENABLE SINGLE MICRO-INSTRUCTION USER MODE.
                                  START IN CONTINUE WITH VALID ONE MICRO
                                  INSTRUCTION T  RESULTS. (DES FALT THE
                                  MICRO-PROCESSOR).

'CLEAR STEP'
                -DISABLE SINGLE MICRO-INSTRUCTION STEP MODE.

'RECOVR'                         -RECOVER TO THE CURRENT PROGRAM

'DUMP <FILE-NAME>'               -DUMP SPECIFIED FILE OR FLOPPY DRIVE 1
'DUMP <DISK-DRIVE#>'             -DUMP SPECIFIED FILE ON FLOPPY DRIVE 1
      NOTE:  'DUMP' IS USED T  SPECIFY A FILE CONTAINING MS MICRO-CODE
             CORRECTED VERSION.  WHEN AN MCS POSITION IN THE CURRENT STORE.
             ADDRESS-N IS DIFFERENT N OF IS THE CURRENT STORE.
             THIS F  AM FILE AN USED FOR ALL VERSIONS OF THE MCS.
             DUMP  MS MCS IS NOT DIRECTLY PROGRAMMED.
```

ERROR MESSAGE HELP FILE

```
?<TEXT-STRING> IS INCOMPLETE
        THE <TEXT-STRING> IS NOT A COMPLETE CONSOLE COMMAND
?<TEXT-STRING> IS INCORRECT
        THE <TEXT-STRING> IS NOT RECOGNIZED AS PART OF A COMMAND
?FILE NAME ERR
        A <FILE-NAME> GIVEN WITH A COMMAND CAN NOT BE TRANSLATED TO RADOS
?FILE NOT FOUND
        A <FILE-NAME> GIVEN WITH A 'LOAD' OR 'DIR' COMMAND DOES NOT
        MATCH ANY FILE ON THE CURRENTLY LOADED FLOPPY DISC, CAN ALSO
        BE GENERATED BY 'HELP','BOOT', IF AN ATTEMPTED ACS LOAD OF
        HELP FILE, BOOT FILE, OR ACS FILE IS MISSING FROM FLOPPY.
?I/O CPU RESPONSE
        CONSOLE FIELD-OUT FAILED FOR A RESPONSE FROM CPU
?CPU NOT IN CONSOLE WAIT LOOP, COMMAND REJECTED
        A CONSOLE COMMAND REQUIRING ASSISTANCE FROM CPU WAS ISSUED
        WHILE THE CPU WAS NOT IN THE CONSOLE SERVICE LOOP
?CPU CLK STOP, COMMAND ABORTED
        A CONSOLE COMMAND THAT REQUIRES THE CPU CLOCK TO BE RUNNING
        WAS ISSUED WHILE THE CLOCK WAS STOPPED
?FLOPPY ERR:CODE=X
        THE CONSOLE FLOPPY DRIVER DETECTED AN ERROR. CODES ARE AS
        FOLLOWS:(CODES ARE ALWAYS PRINTED IN HEX RADIX)
        CODE=1  FLOPPY HARDWARE ERROR(CRC,/ANALYZE/)
        CODE=2  FILE NOT FOUND
        CODE=3  FLOPPY DRIVER QUEUE OVERFLOW
        CODE=4  [illegible] SOFTWARE REQUIRES AN [illegible] SECTOR NUMBER
?FLOPPY BUT HEAD
        THE CONSOLE FLOPPY DRIVE FAILED TO BECOME READY WHEN REQUIRED.
?NO BOOT ON FLOPPY
        CONSOLE ATTEMPTED TO BOOT FROM A FLOPPY THAT DOES NOT CONTAIN
        A VALID BOOT BLOCK.
?FLOPPY ERROR ON BOOT
        A FLOPPY ERROR WAS DETECTED WHILE ATTEMPTING A CONSOLE BOOT.
```

?ASC-ERR ON FUNCTION

A MICRO-ERROR OCCURRED IN THE CPU WHILE OBTAINING A COMMAND
RESPONSE. SEE ERROR REGISTERS AND DUMPED AFTER THIS MESSAGE
IS PRINTED.

?INT-REG ERR

A MICRO-ERROR OCCURRED WHILE ATTEMPTING TO REFERENCE A CPU
INTERNAL(PROCESSOR) REGISTER. AL COLLECT PROCESS WILL CAUSE
THIS ERROR.

?MICRO-ERR, CODE=X

AN UNRECOGNIZED MICRO-ERROR OCCURRED. THE CODE RETURNED BY
THE CPU IS NOT IN THE RANGE OF RECOGNIZED ERROR CODES.
'X' IS THE CODE THAT WAS RETURNED BY THE CPU.

?INT-STK ILLD

THE CPU HALTED BECAUSE THE INTERRUPT STACK WAS MARKED INVALID

?CPU DBLE-FAU HLT

THE CPU HAS DONE A 'DOUBLE ERROR' HALT

?ILL I/O SEC

THE CPU DETECTED AN ILLEGAL INTERRUPT/EXCEPTION VECTOR

?NO USR VCA

CPU RECEIVED AN INTERRUPT/EXCEPTION VECTOR TO USER VCA AND
NO USER HAS EXISTS

?CHK SAR

A CHANGE MODE INSTRUCTION WAS ATTEMPTED FROM THE INTERRUPT STACK

?PAT-FAR FAULT,CODE=XX

A VIRTUAL ADDRESS OR DEPOSIT CAUSED AN ERROR IN THE PEPLAC
MANAGEMENT MICRO-ROUTINES. 'XX' IS A ONE BYTE ERROR CODE
RETURNED BY THE ROUTINE, WITH THE FOLLOWING BIT ASSIGNMENTS:

BIT 0 = LENGTH VIOLATION(SEE NUMBERED ERRORS BELOW)
BIT 1 = FAULT WAS ON A PTE REFERENCE
BIT 2 = WRITE OR MODIFY INTENT
BIT 3 = ACCESS VIOLATION
BITS 4 THRU 7 SHOULD BE ZEROED

?CMD-CON ERR

THE CONSOLE DETECTED BY ERROR IN THE FORMAT OF AN INDIRECT
COMMAND FILE, POSSIBLE NESTING ARGS IS TOO THAN 10 C-ARGUMENTS
IN AN INDIRECT COMMAND LINE, OR A COMMAND LINE DID NOT
END WITH A CARRIAGE RETURN AND LINE FEED.

?AT PENDING

THIS IS NOT ACTUALLY AN ERROR, BUT INDICATES THAT AT LEAST
ONE PENDING AT THE TIME THAT A CONTROL-C/CONTROL
WAIT WAS PERFORMED. CONTINUE CPU TO CLEAR INTERRUPT.

?WARNING-PCS & FPUA VER MISMATCH

THE MICROCODE IN PCS IS NOT COMPATIBLE WITH FPUA.

THIS MESSAGE IS PRINTED ON EACH TOP START OR CONTINUE.DO
NO OTHER ACTION NEED BE CONSOLE.

?FATAL-PCS & PCS VER MISMATCH

THE MICROCODE IN PCS IS NOT COMPATIBLE WITH THAT IN PCS.

TOP START AND CONTINUE ARE DISABLED BY CONSOLE.

?MICRO-MACHINE TIME OUT

INDICATES THAT THE VAX-11/780 MICRO-MACHINE HAS FAILED
TO MINIMUM RESPONSE WITHIN THE MAX TIME PERIOD ALLOWED.

?REMOTE ACCESS NOT SUPPORTED

REMOTE MODE CONSOLE MODE SWITCH ENTERS A 'REMOTE' POSITION,
AND THE REMOTE SUPPORT SOFTWARE IS NOT INCLUDED IN THE CONSOLE.

?SYSAP-6, RESTARTING CONSOLE

THE CONSOLE TOOK A TIME-OUT ERROR, CONSOLE WILL RESTART.

?UNEXPECTED TRAP

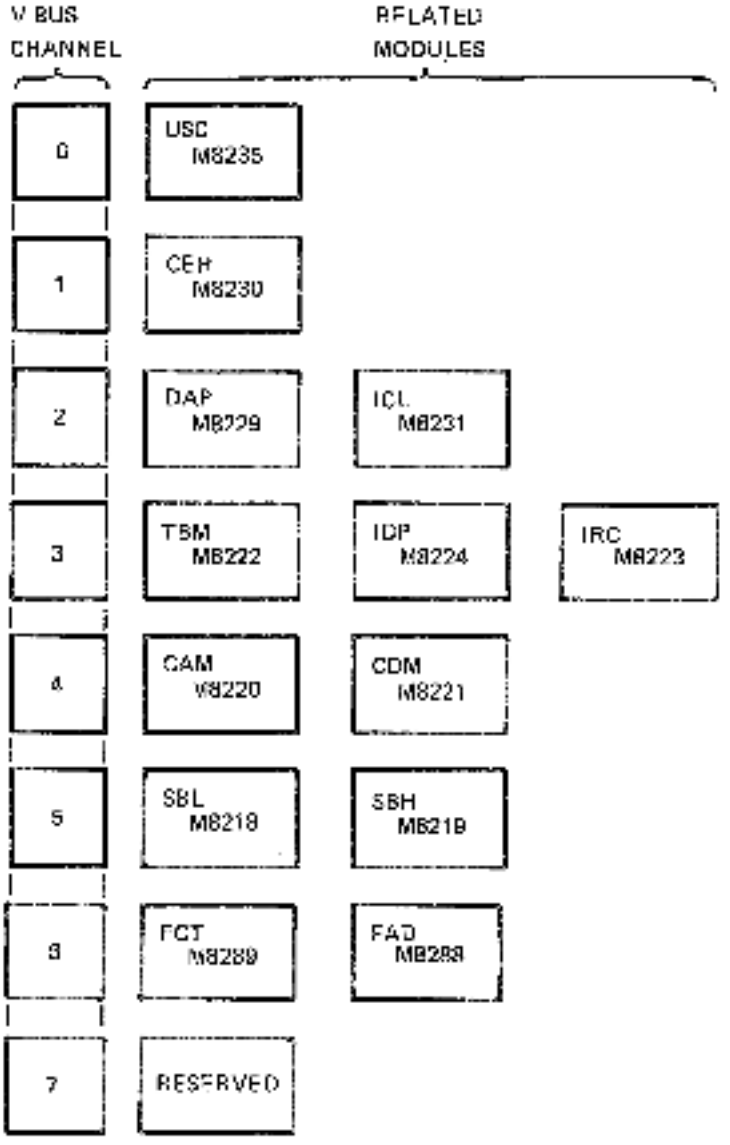
MOUNT CONSOLE FLOPPY, THEN TYPE ^C

CONSOLE TRAPPED TO AN UNUSED VECTOR, CONSOLE RESOURCES MAY BE LOST

?U-BLAST

CONSOLE'S TERMINAL OUTPUT FLOW IS BLOCKED, CONSOLE WILL RESTART.

V BUS CHANNEL CONFIGURATION



```
CHAR    DEC     BIN     HEX     OCTAL   VAL     SIGNAL NAME

00      0000    00000000 00     000     CPIN    UNDF UPCSV 00 H
01      0001    00000001 01     001     CPIN    UNDF UPCSV 01 H
02      0002    00000010 02     002     CPIN    UNDF UPCSV 02 H
03      0003    00000011 03     003     CPIN    UNDF UPCSV 03 H
04      0004    00000100 04     004     CPTA    UNDF UPCSV 04 H
05      0005    00000101 05     005     CPIN    UNDF UPCSV 05 H
06      0006    00000110 06     006     CPTA    UNDF UPCSV 06 H
07      0007    00000111 07     007     CPTN    UNDF UPCSV 07 H
08      0008    00001000 08     010     CPIA    UNDF UPCSV 08 H
09      0009    00001001 09     011     CPTM    UNDF UPCSV 09 H
0A      0010    00001010 0A     012     CPIN    UNDF UPCSV 10 H
0B      0011    00001011 0B     013     CPTB    UNDF UPCSV 11 H
0C      0012    00001100 0C     014     CPIM    UNDF UPCSV 12 H
0D      0013    00001101 0D     015     CPTX    LNCM STALL H
0E      0014    00001110 0E     016     CPIX    UNCM UTRAP H
0F      0015    00001111 0F     017     CPTX    LNCJ ECC 010P41E- 04 H
10      0016    00010000 10     020     CPTI    UNCE ID BUS XCV2 04 L
11      0017    00010001 11     021     CPTS    UNCE CB MP (310721 H
12      0018    00010010 12     022     CPTS    UNCE CB MP (000332) H
13      0019    00010011 13     023     CPTS    UNCE CB MP (000330) H
14      0020    00010100 14     024     CPTX    UNCE NCB MP CYCLE *
15      0021    00010101 15     025     CPTI    UNCM NCB MEM AVAIL L
16      0022    00010110 16     026     CPTX    FETA NCC OVERRIDE *
17      0023    00010111 17     027     CPIX    UNCP ERLP EN (000020) L
18      0024    00011000 18     030     CPTX    /
19      0025    00011001 19     031     CPTX    IELA NLL E ILL H
1A      0026    00011010 1A     032     CPTX    DCPA UNCP H
1B      0027    00011011 1B     033     CPTX    D
1C      0028    00011100 1C     034     CPTX    E
1D      0029    00011101 1D     035     CPTX    F
1E      0030    00011110 1E     036     CPTX    ACCA UNCP *
1F      0031    00011111 1F     037     CPTX    J
20      0032    00100000 20     040     CPTX    K
21      0033    00100001 21     041     CPTX    CEHP FSL E SCP H
22      0034    00100010 22     042     CPTX    IELP A,J E ILL H
23      0035    00100011 23     043     CPTX    M
24      0036    00100100 24     044     CPTX    P
25      0037    00100101 25     045     CPTX    ACCA UNCP *
26      0038    00100110 26     046     CPTX    Q
27      0039    00100111 27     047     CPTX    Y
28      0040    00101000 28     050     CPTX    DCPA LNM1 H
29      0041    00101001 29     051     CPTX    V
2A      0042    00101010 2A     052     CPTX    X
2B      0043    00101011 2B     053     CPTX    Y
2C      0044    00101100 2C     054     CPTX    ACCA UB2 H
2D      0045    00101101 2D     055     CPTX    CEHP UTRAP VECT 2 H
2E      0046    00101110 2E     056     CPTX    FPWD LAST REF ERRS 1 H
2F      0047    00101111 2F     057     CPTX    FAPD BS(1) H
30      0048    00110000 30     060     CPTX    30
31      0049    00110001 31     061     CPTX    CEHC UTRAP VECT 1 *
32      0050    00110010 32     062     CPTX    FPWD LAST REF ERRS 0 H
33      0051    00110011 33     063     CPTX    ICPS BC H,L, 0 H
34      0052    00110100 34     064     CPTX    JJ
```

V BUS DIRECTORY

CHAN	BIT (HEX)	BIT (OCTAL)	BYTE	MODULE	I/O	SIGNAL NAME
00	0000	000001	U1CH	MB235	CPTX	SENE UTRAP VECT 0 H
01	0001	000002	U1CH	MB235	CPTX	SENF MEMIED ERR (1) H
02	0002	000004	U1CH	MB235	CPTX	ICLK KALU Z (1) H
03	0003	000010	U1CH	MB235	CPTX	VV
04	0004	000021	U1CH	MB235	CPTX	SENE UTRAP VECT 3 H
05	0005	000042	U1CH	MB235	CPTX	CKNH FPU WIT L
06	0006	000103	U1CH	MB235	CPTX	ICLK KALU H (1) H
07	0007	000104	U1CH	MB235	CPTX	FT
08	0008	000205	U3CH	MB235	CPTX	U3CH REH ZH (3:8:8) H
09	0009	000206	U3CH	MB235	CPTX	U3CH REH EH (3:8:15) H
0A	000A	000407	U3CH	MB235	CPTX	U3CH REH ZH (3:8:12) H
0B	000B	000410	U3CH	MB235	CPTX	U3CH REH EH (3:8:7C) H
0C	000C	000611	U3CH	MB235	CPTX	U3CH REH EH 72*(8:4) H
0D	000D	000612	U3CH	MB235	CPTX	U3CH SEH EH 72*(8:4) H
0E	000E	000713	U3CH	MB235	CPTX	ICLK SEH[7:0][9:4] H
0F	000F	000714	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
10	0010	001015	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
11	0011	001016	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
12	0012	001117	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
13	0013	001120	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
14	0014	001221	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
15	0015	001222	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
16	0016	001323	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
17	0017	001324	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
18	0018	001425	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
19	0019	001426	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
1A	001A	001527	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
1B	001B	001530	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
1C	001C	001631	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
1D	001D	001632	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
1E	001E	001733	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
1F	001F	001734	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
20	0020	002035	U3CH	MB235	CPTX	U3CH SEH[7:0][8:4] H
21	0021	002036	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
22	0022	002137	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
23	0023	002140	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
24	0024	002241	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
25	0025	002242	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
26	0026	002343	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
27	0027	002344	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
28	0028	002445	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
29	0029	002446	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
2A	002A	002547	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
2B	002B	002550	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
2C	002C	002651	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
2D	002D	002652	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
2E	002E	002753	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
2F	002F	002754	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
30	0030	003055	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
31	0031	003056	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
32	0032	003157	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
33	0033	003160	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
34	0034	003261	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
35	0035	003262	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
36	0036	003363	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
37	0037	003364	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
38	0038	003465	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
39	0039	003466	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
3A	003A	003567	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
3B	003B	003570	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
3C	003C	003671	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
3D	003D	003672	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
3E	003E	003773	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
3F	003F	003774	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
40	0040	004035	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
41	0041	004036	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
42	0042	004137	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
43	0043	004140	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
44	0044	004241	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
45	0045	004242	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
46	0046	004343	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
47	0047	004344	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
48	0048	004445	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
49	0049	004446	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
4A	004A	004547	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
4B	004B	004550	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
4C	004C	004651	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
4D	004D	004652	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
4E	004E	004753	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
4F	004F	004754	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
50	0050	005055	U3CH	MB235	CPTX	U3CH SEH[7:0][8:4] H
51	0051	005056	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
52	0052	005157	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
53	0053	005160	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
54	0054	005261	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
55	0055	005262	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
56	0056	005363	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
57	0057	005364	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
58	0058	005465	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
59	0059	005466	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
5A	005A	005567	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
5B	005B	005570	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
5C	005C	005671	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
5D	005D	005672	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
5E	005E	005773	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
5F	005F	005774	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
60	0060	006035	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
61	0061	006036	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
62	0062	006137	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
63	0063	006140	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
64	0064	006241	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
65	0065	006242	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
66	0066	006343	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
67	0067	006344	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
68	0068	006445	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
69	0069	006446	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
6A	006A	006547	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
6B	006B	006550	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
6C	006C	006651	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
6D	006D	006652	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
6E	006E	006753	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
6F	006F	006754	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
70	0070	007055	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
71	0071	007056	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
72	0072	007157	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
73	0073	007160	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
74	0074	007261	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
75	0075	007262	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
76	0076	007363	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
77	0077	007364	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
78	0078	007465	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
79	0079	007466	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
7A	007A	007567	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
7B	007B	007570	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
7C	007C	007671	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H
7D	007D	007672	U3CH	MB235	CPTX	ICLK SEH[7:0][8:4] H
7E	007E	007773	U3CH	MB235	CPTX	ICLK SEH[7:0][F:C] H
7F	007F	007774	U3CH	MB235	CPTX	U3CH SEH[7:0][F:C] H

V BUS DIRECTORY

CHAN	BIT (HEX)	BIT (OCTAL)	I/O	MODULE	T.S.	SIGNAL NAME
01	8000	100000	CE-E	M8215	CPTX	POSE PWR ERR (P5)(6) H
01	4001	040001	CE-E	M8215	CPTX	*PEBC PWR ERR (A5)(3) H
01	0002	000002	CE-E	M8215	CPTX	P(SE PWR ERR (S)(8) H
01	0004	000004	CE-E	M8215	CPTX	SMLP PWR ERR TRAP L
01	0008	000010	CE-E	M8215	CPTX	TBMM PIOT UTRAP L
01	0010	000020	CEMF	M8215	CPTX	CEMF TRO STATE H
01	0020	000040	CEMF	M8215	CPTX	CEMF DEAD PLUS H
01	0040	000100	CEMF	M8215	CPTX	CEMF LOAD STATE H
01	0080	000200	CEMF	M8215	CPTX	CEMF CLR UM007 (1) H
01	0100	000400	CEMF	M8215	CPTX	(CL8 EN TO XEBUS L
02	0001	000001	CEMA	*8215	CPTX	CEPF BMX3L L
02	0002	000002	CEMA	*8215	CPTX	CCPC BMX15 L
02	0004	000004	CEMA	*8215	CPTX	CCPC BMXPT L
02	0008	000010	CEMA	*8215	CPTX	CCPC BMX31 L
02	0010	000020	CEMA	*8215	CPTX	CCPC BMX15 L
02	0020	000040	CEMA	*8215	CPTX	CCPC BMXBT L
02	0040	000100	CEMA	*8215	CPTX	CCPC ALU BUFF3) L
02	0080	000200	CEMA	*8215	CPTX	CCPL ALU CARRY13 L
02	0100	000400	CEMA	*8215	CPTX	CCPL ALU CARRY7 L
02	0200	000400	CEMA	*8215	CPTX	CCPL ALU CARRY3 L
02	0400	001000	CEMA	*8215	CPTX	CCPL ALU BUFF1) L
02	0800	002000	CEMA	*8215	CPTX	CCPL ALU BUFF16 L
02	1000	004000	CEMA	*8215	CPTX	CCPL ALU BUFF15 L
02	2000	010000	CEMA	*8215	CPTX	CCPL ALU BUFF07 L
02	4000	020000	CEMA	*8215	CPTX	DCPV ALUC3(1)=B L
02	8000	040000	CE-A	M8215	CPTX	DCPF ALUC(7)=B L
03	0001	000001	CE-A	M8215	CPTX	DCPF ALUC3)=B L
03	0002	000002	CE-A	M8215	CPTX	BUS ALU BYTE1 =B H
03	0004	000004	CE-A	M8215	CPTX	BUS ALU BYTE3 =B H
03	0008	000010	CEMF	M8215	CPTX	BUS ALU BYTE3 =B H
03	0010	000020	CEMF	M8215	CPTX	DCPA ANXBP L
03	0020	000040	CEMF	*8215	CPTX	DAPB ALULU=A F,LS H L
03	0040	000100	CEMA	M8215	CPTX	DAPH ALULU=A MSAUB 0 L
03	0080	000200	CEMF	M8215	CPTX	DDPH EALU0B H
03	0100	000400	CEMF	M8215	CPTX	DDPH EALU0B H
03	0200	001000	CEMF	M8215	CPTX	ACCX MDATA H
03	0400	002000	CEMF	*8214	CPTX	ACCX ZDATA H
03	0800	004000	CEMF	*8215	CPTX	ACCX VDATA H
03	1000	010000	CEMF	*8215	CPTX	ACCX CDATA H
03	2000	020000	CEMD	*8215	CPTX	CEMD SECOND REF H
03	4000	040000	CEMD	*8214	CPTX	SBLT STALL L
03	8000	100000	CEMD	*8214	CPTX	IRCJ NO CONT H
04	0001	000001	CEMD	*8215	CPTX	IRCJ FLCAY H
04	0002	000002	CEMD	M8215	CPTX	IRCJ NORD CONT H
04	0004	000004	CEMD	M8215	CPTX	IRCJ BYTE CONT H
04	0008	000010	CEMD	M8215	CPTX	IRCJ SAVE CONTEXT H
04	0010	000020	CEMF	M8215	CPTX	IRCJ FLCAY NZERO H
04	0020	000040	CEMF	M8215	CPTX	USLB CLR UTRAP L
04	0040	000100	CE-A	M8215	CPTX	DCPA *8215(1) H
04	0080	000200	CE-A	M8215	CPTX	DCPJ *8215(1) H
04	0100	000400	CE-A	M8215	CPTX	DCPJ *8215(3) -
04	0200	001000	CE-E	M8215	CPTX	TBMM EN CMDDRIVE H
04	0400	002000	CEMF	M8215	CPTX	TBMM PAGE EDGE H
04	0800	004000	CEME	*8215	CPTX	TBMM EN UNALIGN TRAP H
04	1000	010000	CEMC	M8215	CPTX	SBLP TIMEOUT TRAP L
04	2000	020000	CEME	*8215	CPTX	SBLP PDS TRAP L
04	4000	040000	CEME	*8215	CPTX	TBMM TB PAR UTRAP L
04	8000	100000	CEMF	*8215	CPTX	TBMM MTBS UTRAP L
05	0001	000001	CEME	*8215	CPTX	TBMM MUIT UTRAP L
05	0002	000002	CEME	*8215	CPTX	CEMF CB PF TRAP L
05	0004	000004	CEMD	*8215	CPTX	RESERVED
05	0008	000010	CEMC	*8215	CPTX	RESERVED
05	0010	000020	CEMX	M8215	CPTX	RESERVED
05	0020	000040	CEPX	M8215	CPTX	RESERVED

V BUS DIRECTORY

CHAN	BIT (HEX)	KEY (OCTAL)	D-C	MODULE	T,E,	SIGNAL NAME
02	0000	00000	DAPA	M8229	CPTX	IRQF CPC0 H
02	0001	00001	DAPA	M8229	CPTX	IRQF CPC1 H
02	0002	00002	DAPA	M8229	CPTX	IRQF CPC2 H
02	0003	00003	DAPA	M8229	CPTX	IRQF CPC3 H
02	0004	00004	DAPA	M8229	CPTX	IRQF CPC4 H
02	0005	00005	DAPA	M8229	CPTX	IRQF CPC5 H
02	0006	00006	DAPA	M8229	CPTX	IRQF CPC6 H
02	0007	00007	DAPA	M8229	CPTX	IRQF CPC7 H
02	0008	00010	DAPX	M8229	CPTX	CEMU MEMREF DTACK H
02	0009	00011	DAPX	M8229	CPTX	CEMU MEMREF DISABLE H
02	000A	00012	DAPX	M8229	CPTX	RESERVED
02	000B	00013	DAPX	M8229	CPTX	RESERVED
02	000C	00014	DAPX	M8229	CPTX	IRQF BYTE INT H
02	000D	00015	DAPX	M8229	CPTX	IRQF WORD INT H
02	000E	00016	DAPX	M8229	CPTX	IRQF LPDI INT H
02	000F	00017	DAPX	M8229	CPTX	CPCH PE RES H
02	0010	00020	DAPX	M8229	CPTX	RESERVED
02	0011	00021	DAPX	M8229	CPTX	IRQE SPI CONZ H
02	0012	00022	DAPX	M8229	CPTX	IRQE SPI CONI H
02	0013	00023	DAPX	M8229	CPTX	IRQE SPI CONB H
02	0014	00024	DAPX	M8229	CPTX	DMAE ALCO UPDATE H
02	0015	00025	DAPX	M8229	CPTX	CEMP RESO RESR H
02	0016	00026	DAPX	M8229	CPTX	RESERVED
02	0017	00027	DAPX	M8229	CPTX	RESERVED
02	0018	00030	DAPX	M8229	CPTX	RESERVED
02	0019	00031	DAPX	M8229	CPTX	RESERVED
02	001A	00032	DAPX	M8229	CPTX	RESERVED
02	001B	00033	DAPL	M8229	CPTX	IOPA SPI ADR0 L
02	001C	00034	DAPL	M8229	CPTX	IOPA SPI ADR1 L
02	001D	00035	DAPL	M8229	CPTX	IOPA SPI ADR2 L
02	001E	00036	DAPL	M8229	CPTX	IOPA SPI ADR3 L
02	001F	00037	DAPL	M8229	CPTX	IOPA SP2 ADR0 L
02	0020	00040	DAPL	M8229	CPTX	IOPA SP2 ADR1 L
02	0021	00041	DAPL	M8229	CPTX	IOPA SP2 ADR2 L
02	0022	00042	DAPL	M8229	CPTX	IOPA SP2 ADR3 L
02	0023	00043	DAPL	M8229	CPTX	IOPA PRN 0 L
02	0024	00044	DAPL	M8229	CPTX	IOPA PRN 1 L
02	0025	00045	DAPL	M8229	CPTX	IOPA PRN 2 L
02	0026	00046	DAPL	M8229	CPTX	IOPA PRN 3 L
02	0027	00047	DAPX	M8229	CPTX	RESERVED
02	0028	00050	IOLP	M8231	CPTX	MEBE MEB SYS PAR H
02	0029	00051	IOLP	M8231	CPTX	BALM TIMO CHF INTR L
02	002A	00052	IOLP	M8231	CPTX	SBAL FAULT PATH H
02	002B	00053	IOLP	M8231	CPTX	SBAF SBI ALERT H H
02	002C	00054	IOLP	M8231	CPTX	BALM INT MCO INTR L
02	002D	00055	IOLP	M8231	CPTX	SBAH IOPP INTR H
02	002E	00056	IOLP	M8231	CPTX	SBAH SSI MERT P H
02	002F	00057	IOLP	M8231	CPTX	SBAH SSI REQ6 R H
02	0030	00060	IOLP	M8231	CPTX	SBAH SSI REQ5 R H
02	0031	00061	IOLP	M8231	CPTX	SBAH SSI REQ4 R H
02	0032	00062	IOLP	M8231	CPTX	IOLP CPL ACT L H
02	0033	00063	IOLP	M8231	CPTX	IOLP CPL ACT 1 H
02	0034	00064	IOLP	M8231	CPTX	IOLP CPL ACT 2 H

V BUS DIRECTORY

CHIP	SET (HEX)	SET (BINARY)	BUS	MODULE	TYPE	SIGNAL NAME
M2	0A33	0000 1010 0011 0011	ICLR	M8231	CPTX	ICLR IPL SET L H
M2	0A34	0000 1010 0011 0100	ICLR	M8231	CPTX	ICLR IPL SET 2 H
M2	0A37	0000 1010 0011 0111	ICLR	M8231	CPTX	ICLR PRIOR 5 -
M2	0A38	0000 1010 0011 1000	ICLR	M8231	CPTX	ICLR PRIOR 4 H
M2	0A39	0000 1010 0011 1001	ICLR	M8231	CPTX	ICLR PRIOR 3 H
M2	0A3A	0000 1010 0011 1010	ICLR	M8231	CPTX	ICLR PRIOR 2 H
M2	0A3B	0000 1010 0011 1011	ICLR	M8231	CPTX	ICLR PRIOR 1 H
M2	0A3C	0000 1010 0011 1100	ICLR	M8231	CPTX	ICLR PRIOR 0 H
M2	0A3D	0000 1010 0011 1101	ICLR	M8231	CPTX	ICLR CHTP RED L
M2	0A3E	0000 1010 0011 1110	ICLR	M8231	CPTX	CHGL PDN 3VPA H
M2	0A3F	0000 1010 0011 1111	ICLR	M8231	CPTX	CHGL VMIT 3VPA H
M2	0A40	0000 1010 0100 0000	ICLR	M8231	CPTX	CHGL TRAP CODE2 CL H
M2	0A41	0000 1010 0100 0001	ICLR	M8231	CPTX	CHGL TRAP CODE1 CL H
M2	0A42	0000 1010 0100 0010	ICLR	M8231	CPTX	CHGL TRAP CODE0 CL H
M2	0A43	0000 1010 0100 0011	ICLR	M8231	CPTX	CHGL 1030 H
M2	0A44	0000 1010 0100 0100	ICLR	M8231	CPTX	IRQI STALL+BVC L
M2	0A45	0000 1010 0100 0101	ICLR	M8231	CPTX	CJDM +ALT DFB H
M2	0A46	0000 1010 0100 0110	ICLR	M8231	CPTX	PFEF+FC
M2	0A47	0000 1010 0100 0111	ICLR	M8231	CPTX	DDPB BA+CH3 H
M2	0A48	0000 1010 0100 1000	ICLR	M8231	CPTX	DRPV 09 BIT3 H
M2	0A49	0000 1010 0100 1001	ICLR	M8231	CPTX	DRPV 08 BIT2 H
M2	0A4A	0000 1010 0100 1010	ICLR	M8231	CPTX	DDPB BA+CH-2 H
M2	0A4B	0000 1010 0100 1011	ICLR	M8231	CPTX	DRPV 09 BIT3 H
M2	0A4C	0000 1010 0100 1100	ICLR	M8231	CPTX	DDPB BA+CH-1 H
M2	0A4D	0000 1010 0100 1101	ICLR	M8231	CPTX	DRPV 08 BIT2 H
M2	0A4E	0000 1010 0100 1110	ICLR	M8231	CPTX	DDPB BA+CH-0 -
M2	0A4F	0000 1010 0100 1111	ICLR	M8231	CPTX	DRPV 07 BIT1 -
M2	0A50	0000 1010 0101 0000	ICLR	M8231	CPTX	IRQH RPCT H
M2	0A51	0000 1010 0101 0001	ICLR	M8231	CPTX	IRQH RPCM H
M2	0A52	0000 1010 0101 0010	ICLR	M8231	CPTX	IRQH IPC H H
M2	0A53	0000 1010 0101 0011	ICLR	M8231	CPTX	IRQH READ DP H
M2	0A54	0000 1010 0101 0100	ICLR	M8231	CPTX	PEEF+FC
M2	0A55	0000 1010 0101 0101	ICLR	M8231	CPTX	ICLR REP REMX 52 -
M2	0A56	0000 1010 0101 0110	ICLR	M8231	CPTX	ICLF REP REMX 83 -
M2	0A57	0000 1010 0101 0111	ICLR	M8231	CPTX	ICLR REP REMX 58 -
M2	0A58	0000 1010 0101 1000	ICLH	M8231	CPTX	ICLH 10 FC PBL H
M2	0A59	0000 1010 0101 1001	ICLH	M8231	CPTX	TCLH 10 10 VECT L
M2	0A5A	0000 1010 0101 1010	ICLH	M8231	CPTX	LCLK 10 FC CES H
M2	0A5B	0000 1010 0101 1011	ICLH	M8231	CPTX	ICLH 10 FC 4THP L
M2	0A5C	0000 1010 0101 1100	ICLH	M8231	CPTX	ICLH 10 FC 3THP L
M2	0A5D	0000 1010 0101 1101	ICLH	M8231	CPTX	ICLH 10% 32 L
M2	0A5E	0000 1010 0101 1110	ICLH	M8231	CPTX	ICLH 10% 31 L
M2	0A5F	0000 1010 0101 1111	ICLH	M8231	CPTX	ICLH 10% 30 L
M2	0A60	0000 1010 0110 0000	ICLH	M8231	CPTX	ICLH 10% 30 L
M2	0A61	0000 1010 0110 0001	ICLJ	M8231	CPTX	DDPP BCPS CL1 H
M2	0A62	0000 1010 0110 0010	ICLJ	M8231	CPTX	DDPA SCP4 CL1 H
M2	0A63	0000 1010 0110 0011	ICLJ	M8231	CPTX	DDPR SCP3 CL1 H
M2	0A64	0000 1010 0110 0100	ICLJ	M8231	CPTX	DDPA SCP2 CL1 -
M2	0A65	0000 1010 0110 0101	ICLJ	M8231	CPTX	DDPA SCP1 CL1 -
M2	0A66	0000 1010 0110 0110	ICLJ	M8231	CPTX	DDPA SCP0 CL1 -
M2	0A67	0000 1010 0110 0111	ICLJ	M8231	CPTX	ICLJ D TO 32 L
M2	0A68	0000 1010 0110 1000	ICLJ	M8231	CPTX	ICLJ LD TO 1 L
M2	0A69	0000 1010 0110 1001	ICLJ	M8231	CPTX	PSEN HALUMP H
M2	0A6A	0000 1010 0110 1010	ICLJ	M8231	CPTX	DDPH HALUMP L
M2	0A6B	0000 1010 0110 1011	ICLJ	M8231	CPTX	RESET+EC

V BUS DIRECTORY

SYMBOL	BIT (HEX)	BIT (OCTAL)	DIR	MODULE	I.O.	DESCRIPTION
03	0002	0000	INP	M8222	CPYX	SYNC C TO MD L
04	0004	0000	INP	M8222	CPTX	SYNC MASK TO MD L
05	0008	0000	INP	M8222	CPTX	SYNC MASK TO DRIVERS L
06	0010	0000	INP	M8222	CPTM	SYNC CPTM L
07	0020	0000	INP	M8222	CPTX	SYNC SRP A MP L
08	0040	0000	INP	M8222	CPTX	SYNC SRP B MP L
09	0080	0000	INP	M8222	CPTX	SYNC TO SRP A MATCH H
0A	0100	0000	INP	M8222	CPTX	SYNC TO SRP B MATCH H
0B	0200	0000	INP	M8222	CPTX	SYNC CMODD ADRS TRAP L
0C	0400	0000	INP	M8222	CPTX	SYNC PAGE TRAP H
0D	0800	0000	INP	M8222	CPTX	SYNC CB PAR ERR H
0E	1000	0000	INP	M8222	CPTX	RESERVED
0F	2000	0000	INP	M8222	CPTX	LOCAL ABORT SYNC H
10	4000	0000	INP	M8222	CPTX	SYNC DB M5315 H
11	8000	0000	INP	M8222	CPTX	RESERVED
12	0001	0001	INP	M8222	CPTX	TRAP PAGE L
13	0002	0002	INP	M8222	CPTX	BUS SRP B MP L
14	0004	0004	INP	M8222	CPTX	BUS SRP A MP L
15	0008	0010	INP	M8222	CPTX	TRAP ENABLE CA H
16	0010	0020	INP	M8222	CPTX	RESERVED
17	0020	0040	INP	M8222	CPTX	RESERVED
18	0040	0100	INP	M8222	CPTX	BUS IS SRP LTH H
19	0080	0200	INP	M8222	CPTX	BUS 3 SRP L
1A	0100	0400	INP	M8222	CPTX	RESERVED
1B	0200	0000	INP	M8222	CPTX	RESERVED
1C	0400	0000	INP	M8222	CPTX	RESERVED
1D	0800	0000	INP	M8222	CPTX	CAMP MODIFY L
1E	1000	0000	INP	M8222	CPTX	CAMP PROTECT CODE 0 L
1F	2000	0000	INP	M8222	CPTX	CAMP PROTECT CODE 1 L
20	4000	0000	INP	M8222	CPTX	CAMP PROTECT CODE 2 L
21	8000	0000	INP	M8222	CPTX	CAMP PROTECT CODE 3 L
22	0001	0000	INP	M8224	CPTA	IOPA PLR BA=7[] H
23	0002	0000	INP	M8224	CPTA	IOPA PLR BA=6[] H
24	0004	0000	INP	M8224	CPTA	IOPA BUF BA=5[] H
25	0008	0000	INP	M8224	CPTB	IOPA PLR BA=4[] H
26	0010	0000	INP	M8224	CPTB	IOPA BUF BA=3[] H
27	0020	0000	INP	M8224	CPTB	IOPA BUF BA=2[] H
28	0040	0000	INP	M8224	CPTB	IOPA BUF BA=1[] H
29	0080	0000	INP	M8224	CPTB	IOPA BUF BA=0[] H
2A	0100	0000	INP	M8224	CPTB	IOPA BUF B1=7[] H
2B	0200	0000	INP	M8224	CPTB	IOPA BUF B1=6[] H
2C	0400	0000	INP	M8224	CPTB	IOPA BUF B1=5[] H
2D	0800	0000	INP	M8224	CPTB	IOPA BUF B1=4[] H
2E	1000	0000	INP	M8224	CPTB	IOPA BUF B1=3[] H
2F	2000	0000	INP	M8224	CPTB	IOPA BUF B1=2[] H
30	4000	0000	INP	M8224	CPTB	IOPA BUF B1=1[] H
31	8000	0000	INP	M8224	CPTB	IOPA BUF B1=0[] H
32	0001	0000	INP	M8224	CPTA	IOPA IRE IA[] H
33	0002	0000	INP	M8224	CPTA	IOPA IRE IB[] H
34	0004	0000	INP	M8224	CPTA	IOPA IRE IC[] H
35	0008	0000	INP	M8224	CPTA	IOPA CLR 2 L

V BUS DIRECTORY

CHAR	DEF (HEX)	BIT (OCTAL)	BUS	MODULE	P.G.	SIGNAL NAME
00	0000	000000	IOPH	M8225	CPYX	IOPE SYNC H
01	0001	000002	IOPA	M8224	CPYX	IOPA SYNC H
02	0002	000004	IOPB	M8224	CPYX	IOPA DBI P H/OC H
03	0003	000006	IOPJ	M8224	CPYX	DMLR IN READ DATA L
04	0004	000010	DBIA	M7320	CPYX	RESERVED
05	0005	000012	IOPC	M8224	CPYX	IOPJ COUNT H
06	0006	000014	IOPJ	M8225	CPYX	IOPJ FLUSH L
07	0007	000016	IOPJ	M8224	CPYX	IOPJ MS VALID H
10	0008	000020	IOPJ	M8224	CPYX	IOPJ SS VALID H
11	0009	000022	IOPJ	M8224	CPYX	IOPJ SP VALID H
12	000A	000024	IOPH	M8228	CPYX	IOPI AC *IOE H
13	000B	000026	IOPH	M8228	CPYX	IOPI BE *IOE L
14	000C	000030	IOPH	M8228	CPYX	IOPI BE *IOH L
15	000D	000032	IOPH	M8228	CPYX	IOPI BE *IOC H
16	000E	000034	IOPH	M8228	CPYX	IOPI CP 3 L
17	000F	000036	IOPH	M8228	CPYX	IOPI CP 2 L
20	0010	000040	IOPL	M8225	CPYX	IOPL IO BUS ACV* 24 L
21	0011	000042	IOPA	M8224	CPYX	RESERVED
22	0012	000044	IOPH	M8224	CPYX	IOPH B DEL14 PC 2 H
23	0013	000046	IOPH	M8224	CPYX	IOPH JA BI* P DEST L
24	0014	000050	IOPH	M8224	CPYX	IOPH B DEL14 PC 1 H
25	0015	000052	IOPH	M8224	CPYX	IOPH B DEL14 PC 0 H
26	0016	000054	IOPH	M8224	CPYX	IOPH VAASL L
27	0017	000056	IOPH	M8224	CPYX	IOPH B DEL14 PS 0 H
30	0018	000060	IOPH	M8228	CPYX	IOPH VA MI H
31	0019	000062	IOPH	M8228	CPYX	IOPH VA ZA H
32	001A	000064	IMCH	M8223	CPYX	IOPH BS IOR L
33	001B	000066	IMCH	M8223	CPYX	IOPH PE *IOB L
34	001C	000070	IMCL	M8223	CPYX	LEVEL *PI *SY L
35	001D	000072	IMCH	M8224	CPYX	RESERVED
36	001E	000074	IMCH	M8223	CPYX	IPEP CB *IMMACE H
37	001F	000076	IMCJ	M8223	CPYX	IMEI SP2 IOH H H
40	0020	000100	IMCJ	M8224	CPYX	IPCC SP2 IOH H H
41	0021	000102	IMCH	M8223	CPYX	IPEN IOH* BI L
42	0022	000104	IMCB	M8224	CPYX	IOLD SERVICE *
43	0023	000106	IMCB	M8223	CPYX	IOLD SERVICE PIT H H
44	0024	000110	IMCB	M8223	CPYX	IOLD SERVICE PIT I H
45	0025	000112	IMCB	M8223	CPYX	IOLD SERVICE CLF 2 H
46	0026	000114	IMCL	M8223	CPYX	IMCC EXEC CT 2 H
47	0027	000116	IMCL	M8223	CPYX	IMCC PUPC CT 1 H
50	0028	000120	IMCL	M8223	CPYX	IMCC EXEC CT 2 H
51	0029	000122	IMCH	M8223	CPYX	RESERVED

VBUS DIRECTORY

CHIP	BIT (HEX)	BIT (OCTAL)	DIR	MODULE	T.S.	STRING NAME
0C	0C00	030002	CAMP	4023H	CPTa	TRAP FORCE ERR 7 L
0C	0001	030021	CAM9	4023H	CPTa	TRAP FORCE ERR 1 L
0C	0002	030022	CAM7	4023H	CPTa	TRAP FORCE ERR 0 L
0C	0003	030023	CAM5	4022H	CPTa	68-0 REV PAR FIELD 3 H
0C	0004	030024	CAM3	4022H	CPTa	68-0 REV PAR FIELD 2 H
0C	0005	030025	CAM1	4022H	CPTa	68-0 REV PAR FIELD 1 H
0C	0006	030026	CAM0	4022H	CPTa	68-0 REV PAR FIELD 0 H
0C	0007	030027	CAM3	4022H	CPTa	CAPA 0C ADR PAR 2 OD H
0C	0008	030010	CAM5	4022H	CPTa	CAPA 0C ADR PAR 1 OD H
0C	0009	030011	CAM7	4022H	CPTa	CAPA 0C ADR PAR 0 OD H
0C	000A	030012	CAM1	4022H	CPTa	CAPA CL ADR PAR 2 OD H
0C	000B	030013	CAM7	4022H	CPTa	CAPA CL ADR PAR 1 OD H
0C	000C	030014	CAM7	4022H	CPTa	CAPA CL ADR PAR 0 OD H
0C	000D	030015	CAM6	4022A	CPTa	CAPA IN PAR 2 H
0C	000E	030016	CAM4	4022H	CPTa	CAPL TO PAR 1 H
0C	000F	030017	CAM2	4022A	CPTa	CAPL TM PAR 0 H
0C	0010	030020	CAM4	4022H	CPTa	CAPK 01 MATCH H
0C	0011	030021	CAM1	4022A	CPTa	CAPK 0R MATCH H
0C	0012	030022	CAM0	4022H	CPTa	CAPA 08[4106 D47? SL H
0C	0013	030023	CAM7	4022H	CPTa	CAPA ERF 4153 D47? S0 H
0C	0014	030024	CAM5	4022A	CPTa	CAPK CPT5 0 H
0C	0015	030025	CAM3	4022H	CPTa	CAPK CPT2 0 H
0C	0016	030026	CAM1	4022H	CPTa	CAPK CPT1 0 L
0C	0017	030027	CAM7	4022H	CPTa	CAPK CPV1 0 H
0C	0018	030028	CAM5	4022H	CPTa	CAPK 01 WRITE 4W41_E H
0C	0019	030031	CAM3	4022A	CPTa	CAPK 0R WRITE 4W41_E H
0C	001A	030032	CAM1	4022A	CPTa	CB-K FORCE 4106 01 H
0C	001B	030033	CAM7	4022H	CPTa	CB-K FORCE 4106 0R H
0C	001C	030034	CAM5	4022A	CPTa	RESERVED
0C	001D	030035	CAM3	4022H	CPTa	CAPE _ATCH VALID 4[1 H
0C	001E	030036	CAM1	4022H	CPTa	TRAP CRCK ERR 0 L
0C	001F	030037	CAM0	4022H	CPTa	SAME 53a PAR FIELD 3 L
0C	0020	030040	CAML	4022A	CPTa	CAML 5[BYTE 3 PAR 00 H
0C	0021	030041	CAML	4022H	CPTa	CAML 5[BYTE 2 PAR 00 H
0C	0022	030042	CAML	4022A	CPTa	CAPL 2[BYTE 1 PAR 00 H
0C	0023	030043	CAML	4022A	CPTa	CAPL 0[BYTE 1 PAR 00 H
0C	0024	030044	CAML	4022A	CPTa	CAPL 5L BYTE A PAR 00 H
0C	0025	030045	CAML	4022A	CPTa	CAPL 2[BYTE B PAR 00 H
0C	0026	030046	CAML	4022H	CPTa	CAPL 09 BYTE 3 PAR 00 H
0C	0027	030047	CAML	4022H	CPTa	CAPL 07 BYTE 2 PAR 00 H
0C	0028	030050	CAML	4022H	CPTa	CAPL 2V BYTE 1 PAR 00 H
0C	0029	030051	CAML	4022H	CPTa	CAPL 07 BYTE 0 PAR 00 H
0C	002A	030052	CAML	4022H	CPTa	CAPL 5[BYTE B PAR 00 H
0C	002B	030053	CAML	4022H	CPTa	CAPL 2C BYTE B PAR 00 H
0C	002C	030054	CAM9	4022A	CPTa	CAPK PAC PAR 2 EVEN H
0C	002D	030055	CAM8	4022H	CPTa	CAPK PAG PAR 1 EVEN H
0C	002E	030056	CAM6	4022A	CPTa	CAPK PAC PAR 0 EVEN H
0C	002F	030057	CAM5	4022A	CPTa	CAPE PA LATCH 12 H
0C	0030	030060	CAM3	4022H	CPTa	CAPE PA LATCH 13 H
0C	0031	030061	CAM2	4022H	CPTa	CAPE PA LATCH 14 H
0C	0032	030062	CAM0	4022A	CPTa	CAPE PA LATCH 15 H
0C	0033	030063	CAM7	4022A	CPTa	CAPE PA LATCH 16 H
0C	0034	030064	CAM5	4022A	CPTa	CAPE PA LATCH 17 H
0C	0035	030065	CAM3	4022H	CPTa	CAPE PA LATCH 18 H
0C	0036	030066	CAM2	4022A	CPTa	CAPE PA LATCH 19 H
0C	0037	030067	CAM0	4022H	CPTa	CAPE PA LATCH PA H

VBUS DIRECTORY

CHIP	BIT (HEX)	BIT (OCTAL)	PKG	MODULE	TYPE	SIGNAL NAME
0x	0x100	0004?0	CPU0	M222R	CPT1	CPUR PA LATCH 21 H
0x	0x39	000?1	CPU0	M222K	CPT1	CPUR PA LATCH 22 H
0x	0x1A	000?2	CPU0	M222P	CPT1	CPUR PA LATCH 23 H
0x	0x0A	000?4	CPU0	M222Q	CPT1	CPUR PA LATCH 24 H
0x	0x06	000?4	CPU0	M222R	CPT1	CPUR PA LATCH 25 H
0x	0x3C	000?4	CPU0	M222S	CPT1	CPUR PA LATCH 26 H
0x	0x3D	000?5	CPU0	M222T	CPT1	CPUR PA LATCH 27 H
0x	0x3E	000?6	CPU0	M222U	CPT1	CPUR PA LATCH 28 H
0x	0x3F	000?7	CPU0	M222V	CPT1	CPUR PA LATCH 29 H
0x	0x40	001?0	CPU0	M2220	CPT2	RESERVED
0x	0x41	001?1	CPU0	M2221	CPT2	RESERVED
0x	0x42	001?2	CPU0	M2222	CPT2	RESERVED
0x	0x43	001?3	CPU0	M2223	CPT2	RESERVED
0x	0x44	001?4	CPU0	M2224	CPT2	RESERVED
0x	0x45	001?5	CPU0	M2225	CPT2	RESERVED
0x	0x46	001?6	CPU0	M2226	CPT2	RESERVED
0x	0x47	001?7	CPU0	M2227	CPT2	CPU0 CPT2 0 H
0x	0x48	002?0	CPU0	M2230	CPT3	CPU0 CPT3 0 L
0x	0x49	002?1	CPU0	M2231	CPT3	CPU0 CPT3 1 H
0x	0x4A	002?2	CPU0	M2232	CPT3	TRVD EN CPU 0x?3 L
0x	0x4B	002?3	CPU0	M2233	CPT3	CPU0 01 A1 PA9 ODD H
0x	0x4C	002?4	CPU0	M2234	CPT3	CPU0 01 A0 PA8 EVEN H
0x	0x4D	002?5	CPU0	M2235	CPT3	CPU0 01 B2 PA5 ODD H
0x	0x4E	002?6	CPU0	M2236	CPT3	CPU0 01 B1 PA4 EVEN H
0x	0x4F	002?7	CPU0	M2237	CPT3	CPU0 01 B0 PA3 ODD H
0x	0x50	003?0	CPU0	M2240	CPT4	CPU0 02 A3 PA2 EVEN H
0x	0x51	003?1	CPU0	M2241	CPT4	CPU0 02 A2 PA1 ODD H
0x	0x52	003?2	CPU0	M2242	CPT4	CPU0 02 A1 PA0 EVEN H
0x	0x53	003?3	CPU0	M2243	CPT4	CPU0 02 A0 PB9 ODD H
0x	0x54	003?4	CPU0	M2244	CPT4	CPU0 02 B3 PB8 EVEN H
0x	0x55	003?5	CPU0	M2245	CPT4	CPU0 02 B2 PB7 ODD H
0x	0x56	003?6	CPU0	M2246	CPT4	CPU0 02 B1 PB6 EVEN H
0x	0x57	003?7	CPU0	M2247	CPT4	CPU0 02 B0 PB5 ODD H
0x	0x58	004?0	CPU0	M2250	CPT5	RESERVED
0x	0x59	004?1	CPU0	M2251	CPT5	RESERVED
0x	0x5A	004?2	CPU0	M2252	CPT5	RESERVED
0x	0x5B	004?3	CPU0	M2253	CPT5	RESERVED
0x	0x5C	004?4	CPU0	M2254	CPT5	CPU0 ADDR LATCH 11 H
0x	0x5D	004?5	CPU0	M2255	CPT5	CPU0 ADDR LATCH 10 H
0x	0x5E	004?6	CPU0	M2256	CPT5	CPU0 ADDR LATCH 9 H
0x	0x5F	004?7	CPU0	M2257	CPT5	CPU0 ADDR LATCH 8 H
0x	0x60	005?0	CPU0	M2260	CPT6	CPU0 ADDR LATCH 7 H
0x	0x61	005?1	CPU0	M2261	CPT6	CPU0 ADDR LATCH 6 H
0x	0x62	005?2	CPU0	M2262	CPT6	CPU0 ADDR LATCH 5 H
0x	0x63	005?3	CPU0	M2263	CPT6	CPU0 ADDR LATCH 4 H
0x	0x64	005?4	CPU0	M2264	CPT6	CPU0 ADDR LATCH 3 H
0x	0x65	005?5	CPU0	M2265	CPT6	CPU0 ADDR LATCH 2 H
0x	0x66	005?6	CPU0	M2266	CPT6	CPU0 ADDR LATCH 1 H
0x	0x67	005?7	CPU0	M2267	CPT6	CPU0 ADDR LATCH 0 H
0x	0x68	006?0	CPU0	M2270	CPT7	68HF REV PAR 3 L
0x	0x69	006?1	CPU0	M2271	CPT7	68HF REV PAR 2 L
0x	0x6A	006?2	CPU0	M2272	CPT7	68HF REV PAR 1 L
0x	0x6B	006?3	CPU0	M2273	CPT7	68HF REV PAR 0 L
0x	0x6C	006?4	CPU0	M2274	CPT7	CAHP 01 WRITE ENABLE H
0x	0x6D	006?5	CPU0	M2275	CPT7	CAHP 02 WRITE ENABLE H
0x	0x6E	006?6	CPU0	M2276	CPT7	RESERVED
0x	0x6F	006?7	CPU0	M2277	CPT7	RESERVED
0x	0x70	007?0	CPU0	M2300	CPT8	CPU1 MASK 1 H
0x	0x71	007?1	CPU0	M2301	CPT8	CPU1 MASK 2 H
0x	0x72	007?2	CPU0	M2302	CPT8	CPU1 MASK 3 H
0x	0x73	007?3	CPU0	M2303	CPT8	CPU1 MASK 4 H
0x	0x74	007?4	CPU0	M2304	CPT8	CPU1 MASK 5 H
0x	0x75	007?5	CPU0	M2305	CPT8	CPU1 MASK 6 H
0x	0x76	007?6	CPU0	M2306	CPT8	CPU1 MASK 7 H
0x	0x77	007?7	CPU0	M2307	CPT8	CPU1 MASK 8 H

INDEX	BIT (HEX)	SET (OCTAL)	DMD	MODULE	PAGE	SIGNAL NAME
85	4000	20000	8000	M8218	CPT4	BUS ED (0) DRIVERS L
86	8001	20001	8001	M8218	CPT4	BUS ED (1) DRIVERS L
87	8002	20002	8002	M8218	CPT4	BUS ED (2) DRIVERS L
88	8003	20003	8003	M8218	CPT4	BUS ED (3) DRIVERS L
89	8004	20004	8004	M8218	CPT4	BUS ED (4) DRIVERS L
8A	8005	20005	8005	M8218	CPT4	BUS ED (5) DRIVERS L
8B	8006	20006	8006	M8218	CPT4	BUS ED (6) DRIVERS L
8C	8007	20007	8007	M8218	CPT4	BUS ED (7) DRIVERS L
8D	8008	20010	8008	M8218	CPT4	BUS ED (8) DRIVERS L
8E	8009	20011	8009	M8218	CPT4	BUS ED (9) DRIVERS L
8F	800A	20012	800A	M8218	CPT4	BUS ED (10) DRIVERS L
90	800B	20013	800B	M8218	CPT4	BUS ED (11) DRIVERS L
91	800C	20014	800C	M8218	CPT4	BUS ED (12) DRIVERS L
92	800D	20015	800D	M8218	CPT4	BUS ED (13) DRIVERS L
93	800E	20016	800E	M8218	CPT4	BUS ED (14) DRIVERS L
94	800F	20017	800F	M8218	CPT4	BUS ED (15) DRIVERS L
95	8010	20020	8010	M8218	CPT4	BUS ED (16) DRIVERS L
96	8011	20021	8011	M8218	CPT4	BUS ED (17) DRIVERS L
97	8012	20022	8012	M8218	CPT4	BUS ED (18) DRIVERS L
98	8013	20023	8013	M8218	CPT4	BUS ED (19) DRIVERS L
99	8014	20024	8014	M8218	CPT4	BUS ED (20) DRIVERS L
9A	8015	20025	8015	M8218	CPT4	BUS ED (21) DRIVERS L
9B	8016	20026	8016	M8218	CPT4	BUS ED (22) DRIVERS L
9C	8017	20027	8017	M8218	CPT4	BUS ED (23) DRIVERS L
9D	8018	20030	8018	M8218	CPT4	BUS ED (24) DRIVERS L
9E	8019	20031	8019	M8218	CPT4	BUS ED (25) DRIVERS L
9F	801A	20032	801A	M8218	CPT4	BUS ED (26) DRIVERS L
A0	801B	20033	801B	M8218	CPT4	BUS ED (27) DRIVERS L
A1	801C	20034	801C	M8218	CPT4	BUS ED (28) DRIVERS L
A2	801D	20035	801D	M8218	CPT4	BUS ED (29) DRIVERS L
A3	801E	20036	801E	M8218	CPT4	BUS ED (30) DRIVERS L
A4	801F	20037	801F	M8218	CPT4	BUS ED (31) DRIVERS L
A5	2220	21000	8020	M8218	CPT5	BUS ED (32) DRIVERS L
A6	2221	21001	8021	M8218	CPT5	BUS ED (33) DRIVERS L
A7	2222	21002	8022	M8218	CPT5	BUS ED (34) DRIVERS L
A8	2223	21003	8023	M8218	CPT5	BUS ED (35) DRIVERS L
A9	2224	21004	8024	M8218	CPT5	BUS ED (36) DRIVERS L
AA	2225	21005	8025	M8218	CPT5	BUS ED (37) DRIVERS L
AB	2226	21006	8026	M8218	CPT5	BUS ED (38) DRIVERS L
AC	2227	21007	8027	M8218	CPT5	BUS ED (39) DRIVERS L
AD	2228	21010	8028	M8218	CPT5	BUS ED (40) DRIVERS L
AE	2229	21011	8029	M8218	CPT5	BUS ED (41) DRIVERS L
AF	222A	21012	802A	M8218	CPT5	BUS ED (42) DRIVERS L
B0	222B	21013	802B	M8218	CPT5	BUS ED (43) DRIVERS L
B1	222C	21014	802C	M8218	CPT5	BUS ED (44) DRIVERS L
B2	222D	21015	802D	M8218	CPT5	BUS ED (45) DRIVERS L
B3	222E	21016	802E	M8218	CPT5	BUS ED (46) DRIVERS L
B4	222F	21017	802F	M8218	CPT5	BUS ED (47) DRIVERS L
B5	2230	21020	8030	M8218	CPT5	BUS ED (48) DRIVERS L
B6	2231	21021	8031	M8218	CPT5	BUS ED (49) DRIVERS L
B7	2232	21022	8032	M8218	CPT5	BUS ED (50) DRIVERS L
B8	2233	21023	8033	M8218	CPT5	BUS ED (51) DRIVERS L
B9	2234	21024	8034	M8218	CPT5	BUS ED (52) DRIVERS L
BA	2235	21025	8035	M8218	CPT5	BUS ED (53) DRIVERS L
BB	2236	21026	8036	M8218	CPT5	BUS ED (54) DRIVERS L
BC	2237	21027	8037	M8218	CPT5	BUS ED (55) DRIVERS L
BD	2238	21030	8038	M8218	CPT5	BUS ED (56) DRIVERS L
BE	2239	21031	8039	M8218	CPT5	BUS ED (57) DRIVERS L
BF	223A	21032	803A	M8218	CPT5	BUS ED (58) DRIVERS L
C0	223B	21033	803B	M8218	CPT5	BUS ED (59) DRIVERS L
C1	223C	21034	803C	M8218	CPT5	BUS ED (60) DRIVERS L
C2	223D	21035	803D	M8218	CPT5	BUS ED (61) DRIVERS L
C3	223E	21036	803E	M8218	CPT5	BUS ED (62) DRIVERS L
C4	223F	21037	803F	M8218	CPT5	BUS ED (63) DRIVERS L
C5	4440	42000	8040	M8218	CPT6	BUS ED (64) DRIVERS L
C6	4441	42001	8041	M8218	CPT6	BUS ED (65) DRIVERS L
C7	4442	42002	8042	M8218	CPT6	BUS ED (66) DRIVERS L
C8	4443	42003	8043	M8218	CPT6	BUS ED (67) DRIVERS L
C9	4444	42004	8044	M8218	CPT6	BUS ED (68) DRIVERS L
CA	4445	42005	8045	M8218	CPT6	BUS ED (69) DRIVERS L
CB	4446	42006	8046	M8218	CPT6	BUS ED (70) DRIVERS L
CC	4447	42007	8047	M8218	CPT6	BUS ED (71) DRIVERS L
CD	4448	42010	8048	M8218	CPT6	BUS ED (72) DRIVERS L
CE	4449	42011	8049	M8218	CPT6	BUS ED (73) DRIVERS L
CF	444A	42012	804A	M8218	CPT6	BUS ED (74) DRIVERS L
D0	444B	42013	804B	M8218	CPT6	BUS ED (75) DRIVERS L
D1	444C	42014	804C	M8218	CPT6	BUS ED (76) DRIVERS L
D2	444D	42015	804D	M8218	CPT6	BUS ED (77) DRIVERS L
D3	444E	42016	804E	M8218	CPT6	BUS ED (78) DRIVERS L
D4	444F	42017	804F	M8218	CPT6	BUS ED (79) DRIVERS L
D5	4450	42020	8050	M8218	CPT6	BUS ED (80) DRIVERS L
D6	4451	42021	8051	M8218	CPT6	BUS ED (81) DRIVERS L
D7	4452	42022	8052	M8218	CPT6	BUS ED (82) DRIVERS L
D8	4453	42023	8053	M8218	CPT6	BUS ED (83) DRIVERS L
D9	4454	42024	8054	M8218	CPT6	BUS ED (84) DRIVERS L
DA	4455	42025	8055	M8218	CPT6	BUS ED (85) DRIVERS L
DB	4456	42026	8056	M8218	CPT6	BUS ED (86) DRIVERS L
DC	4457	42027	8057	M8218	CPT6	BUS ED (87) DRIVERS L
DD	4458	42030	8058	M8218	CPT6	BUS ED (88) DRIVERS L
DE	4459	42031	8059	M8218	CPT6	BUS ED (89) DRIVERS L
DF	445A	42032	805A	M8218	CPT6	BUS ED (90) DRIVERS L
E0	445B	42033	805B	M8218	CPT6	BUS ED (91) DRIVERS L
E1	445C	42034	805C	M8218	CPT6	BUS ED (92) DRIVERS L
E2	445D	42035	805D	M8218	CPT6	BUS ED (93) DRIVERS L
E3	445E	42036	805E	M8218	CPT6	BUS ED (94) DRIVERS L
E4	445F	42037	805F	M8218	CPT6	BUS ED (95) DRIVERS L
E5	6660	63000	8060	M8218	CPT7	BUS ED (96) DRIVERS L
E6	6661	63001	8061	M8218	CPT7	BUS ED (97) DRIVERS L
E7	6662	63002	8062	M8218	CPT7	BUS ED (98) DRIVERS L
E8	6663	63003	8063	M8218	CPT7	BUS ED (99) DRIVERS L
E9	6664	63004	8064	M8218	CPT7	BUS ED (100) DRIVERS L
EA	6665	63005	8065	M8218	CPT7	BUS ED (101) DRIVERS L
EB	6666	63006	8066	M8218	CPT7	BUS ED (102) DRIVERS L
EC	6667	63007	8067	M8218	CPT7	BUS ED (103) DRIVERS L
ED	6668	63010	8068	M8218	CPT7	BUS ED (104) DRIVERS L
EE	6669	63011	8069	M8218	CPT7	BUS ED (105) DRIVERS L
EF	666A	63012	806A	M8218	CPT7	BUS ED (106) DRIVERS L
F0	666B	63013	806B	M8218	CPT7	BUS ED (107) DRIVERS L
F1	666C	63014	806C	M8218	CPT7	BUS ED (108) DRIVERS L
F2	666D	63015	806D	M8218	CPT7	BUS ED (109) DRIVERS L
F3	666E	63016	806E	M8218	CPT7	BUS ED (110) DRIVERS L
F4	666F	63017	806F	M8218	CPT7	BUS ED (111) DRIVERS L
F5	6670	63020	8070	M8218	CPT7	BUS ED (112) DRIVERS L
F6	6671	63021	8071	M8218	CPT7	BUS ED (113) DRIVERS L
F7	6672	63022	8072	M8218	CPT7	BUS ED (114) DRIVERS L
F8	6673	63023	8073	M8218	CPT7	BUS ED (115) DRIVERS L
F9	6674	63024	8074	M8218	CPT7	BUS ED (116) DRIVERS L
FA	6675	63025	8075	M8218	CPT7	BUS ED (117) DRIVERS L
FB	6676	63026	8076	M8218	CPT7	BUS ED (118) DRIVERS L
FC	6677	63027	8077	M8218	CPT7	BUS ED (119) DRIVERS L
FD	6678	63030	8078	M8218	CPT7	BUS ED (120) DRIVERS L
FE	6679	63031	8079	M8218	CPT7	BUS ED (121) DRIVERS L
FF	667A	63032	807A	M8218	CPT7	BUS ED (122) DRIVERS L

PIN #	NET (HEX)	ID (DEFAULT)	PIN	MODULE	TYPE	SIGNAL NAME
01	0035	3F265	3PH1	MB219	CPTX	3PH1 MPITE D4?? 31 H
02	0036	77694	3PH2	MB219	CPTX	3PH2 MPITE D4?? 32 H
03	0037	7D767	3PH3	MB219	CPTX	3PH3 MPITE D4?? 33 H
04	0038	32678	3PH4	MB219	CPTX	3PH4 MPITE D4?? 34 H
05	0039	3E1FL	3PH5	MB219	CPTX	3PH5 MPITE D4?? 35 H
06	003A	7D678	3PH6	MB219	CPTX	3PH6 MPITE D4?? 36 H
07	003B	33275	3PH7	MB219	CPTX	3PH7 MPITE D4?? 37 H
08	003C	77275	3PH8	MB219	CPTX	3PH8 MPITE D4?? 38 H
09	003D	7D279	3PH9	MB219	CPTX	3PH9 MPITE D4?? 39 H
0A	003E	06276	3PHA	MB219	CPTX	3PHA MPITE D4?? 3A H
0B	003F	A7277	3PHB	MB219	CPTX	3PHB MPITE D4?? 3B H
0C	3C4W	M3120	3PH0	MB219	CPTI	3PH0 RECEIVE M454 C H
0D	A731	AA171	3PHA	MB219	CPYI	3PHA RECEIVE M454 L H
0E	3C4E	03113	3PHB	MB219	CPTI	3PHB RECEIVE M454 S H
0F	7933	0B173	3PH9	MB219	CPYI	3PH9 RECEIVE M454 S H
10	3710	M3178	3PHD	MB219	CPTX	3LE MD BYTE 2 PAR H
11	3745	3M385	3PHD	MB219	CPTX	3LE MD BYTE 3 PAR H
12	3716	03134	3PHL	MB219	CPTX	3PHA BUFFER FULL L
13	3747	03137	3PHL	MB219	CPTY	3PLE LATE EXPECT MD L
14	3714	M3110	3PHX	MB219	CPYX	3PLE REC PAR 1 H
15	3744	03111	3PHE	MB219	CPYX	3PLE REC PAR 1 H
16	3716	M3116	3PHE	MB219	CPTX	3PLE REC PAR 2 H
17	3748	03115	3PHE	MB219	CPTY	3PLE REC PAR 3 H
18	3719	M3118	3PHH	MB219	CPTX	3H 3EL 1 L
19	3716	03115	3PHH	MB219	CPTX	3H 3EL 2 L
1A	3716	0A115	3PHH	MB219	CPTX	3H 3EL 4 L
1B	3717	03113	3PHH	MB219	CPTX	3H 3EL 8 L
1C	3719	M3197	3PHP	MB219	CPTX	3PH BUF UNIT A L
1D	3751	03171	3PHP	MB219	CPTX	3PH BUF UNIT 1 L
1E	3752	0M132	3PHP	MB219	CPTX	3PH BUF UNIT 2 L
1F	3713	M3137	3PHP	MB219	CPTX	3PH BUF UNIT 3 L
20	3753	03124	3PHQ	MB219	CPYX	3PH BUF LOOP L
21	3715	M3125	3PHP	MB219	CPTX	3PH BUF UFS L
22	3750	0M126	3PHH	MB219	CPTX	3PH SELECT SA3 #345 L
23	MP17	0A12F	3PHP	MB219	CPTX	32H FRAME ENABLE L
24	3754	03134	3PHD	MB219	CPTX	32H EX SBI DAT4 L
25	A654	03133	3PHF	MB219	CPTX	3PLE FRAME PAR L
26	3733	M3132	3PHR	MB219	CPTX	3PLL TRANSMIT CK H
27	M032	03135	3PHH	MB219	CPTL	CEHH CLR MODE B H
28	3M17	03134	3PHS	MB219	CPTX	CLKL SYS UNIT B L
29	3PA7	03136	3PHP	MB219	CPTX	CEHH CLR MODE 1 H
2A	3P65	03136	3PHP	MB219	CPTX	TRNV CTS PROT L
2B	3P67	03137	3PHX	MB219	CPTX	PLES4HED
2C	3P61	03138	3PHX	MB219	CPTX	PESP4HED
2D	3P61	03141	3PHX	MB219	CPTX	PLBS4HED
2E	3M6P	03142	32HX	MB219	CPTX	PLBE4HED
2F	3P6J	03143	33HX	MB219	CPTX	PLBE4HED
30	3P6K	03144	34HX	MB219	CPTX	PLES4HED
31	3P6E	03145	35HX	MB219	CPTX	PLBE4HED
32	3P6F	03146	36HX	MB219	CPTX	PLBE4HED
33	3P6G	03147	37HX	MB219	CPTX	PLES4HED
34	3P6H	03150	38HX	MB219	CPTX	PLBE4HED
35	3P6J	03151	39HX	MB219	CPTX	PLES4HED
36	3P6K	03152	3AHX	MB219	CPTX	PLBE4HED
37	3P6L	03153	3BHX	MB219	CPTX	PLES4HED
38	3P6M	03154	3CHX	MB219	CPTX	PLBE4HED
39	3P6N	03155	3DHX	MB219	CPTX	PLES4HED
3A	3P6P	03156	3EHX	MB219	CPTX	PLBE4HED
3B	3P6Q	03157	3FHX	MB219	CPTX	PLES4HED

V BUS DIRECTORY

CHAP	REF (HEX)	REF (OCTAL)	DWG	MODULE	TYPE	SIGNAL NAME
2a	0AA0	05037	FCTP	M0206	CPTX	DSPL ACC 04 CONTENT N H
2a	0AC3	05303	FCTP	M0206	CPTX	DSPL ACC 03 CONTENT L H
2c	0A8C	05214	FCTC	M0206	CPTX	FCTC CLR AR L
24	0A00	05000	FCTK	M0206	CPYX	FCTK DP 3=>C H
2d	0A3F	05176	FNHE	M0240	CPTX	FNHE BUS_ERR L
64	0A3A	05035	FCTJ	M0206	CPYX	FCTJ ACC 4 INTR H
6a	2201	02040	FCTC	M0206	CPTX	FCTC ACC 2 INTR H
6d	83A9	20647	FCTL	M0206	CPTX	FCTL ACC 1 INTR H
6e	0A03	05010	FNH1	M0206	CPTS	FCTC RA >DR8 9 L
66	2A1A	05011	FNH1	M0206	CPTS	FCTC RB >DR8 8 L
6e	2A2A	05012	FNH1	M0206	CPT3	FCTC RC >DR8 7 L
66	2A2E	05013	FNH1	M0206	CPT3	FCTC RD >DR8 6 L
6e	2A2C	05014	FNH1	M0206	CPY3	FCTC RE >DR8 5 L
66	2A0C	05015	FNH1	M0206	CPT3	FCTC RH >DR8 4 L
6a	0A0E	05016	FNH1	M0206	CPY3	FCTC RG >DR8 3 L
65	0A0F	05017	FNH1	M0206	CPT3	FCTC RG >DR8 2 L
6a	0A10	05020	SSSS	M0206	CPYX	RESERVED
6e	0A11	05021	SPEA	M0206	CPTX	RESERVED
6a	0A12	05022	FNH1	M0206	CPTX	FCTC CCMPL L
65	0A13	05023	FNH1	M0206	CPTX	FCTC BRC INT H
66	0A15	05025	FNH1	M0252	CPTX	FNH8 EALU CIN L
66	0A16	05026	FNH1	M0252	CPTX	FCTC SEL >CNM H
6a	0A17	05027	FNH1	M0206	CPYX	RESERVED
66	0A18	05030	FNH1	M0252	CPT3	FCTC LC20 BRC H
6a	0A19	05031	FNH1	M0206	CPYX	FCTC LOAC BRL H
6e	0A1A	05032	FNH1	M0206	CPT3	FCTC LOA3 BRC H
6a	0A1B	05033	FNH1	M0206	CPTX	FCTC LOAC BRL H
6e	0A1C	05034	FNH1	M0206	CPT3	FCTC LOA3 BRC H
6a	0A1D	05035	FNHT	M0209	CPTX	FCTC BUS_REQ L
6a	0A1E	05036	FNH1	M0206	CPTX	RESERVED
66	0A1F	05037	FNH1	M0240	CPTX	RESERVED
6e	0A20	05040	FNH1	M0206	CPY1	FCTC ACC4 BR H L
66	0A21	05041	FNH1	M0240	CPT3	FCTC A GT B H
6a	0A22	05042	FNH1	M0253	CPT3	FCTC BHF MUX EN1 L
6b	0A23	05043	FNH1	M0253	CPT3	FCTC BHF MUX EN0 L
6a	0A24	05044	FNH1	M0256	CPY1	FCTC FALU FUNC SEL 2 H
6b	0A25	05045	FNH1	M0256	CPY3	FCTC FALU FUNC SEL 1 H
6a	0A26	05046	FNH1	M0256	CPT3	FCTC FALU FUNC SEL 0 H
6b	0A27	05047	FNH1	M0256	CPT3	FCTC FAMX SEL 1 H
6a	0A28	05050	FNH1	M0226	CPT3	FCTF BHF COUNT 5 H
6b	0A29	05051	FNH1	M0226	CPT3	FCTF BHF COUNT 4 H
6b	0A2A	05052	FNHT	M0226	CPT3	FCTF BHF COUNT 3 H
65	0A2B	05053	FNH1	M0226	CPT3	FCTF BHF COUNT 2 H
6a	0A2C	05054	FNH1	M0226	CPT3	FCTF BHF COUNT 1 H
6a	0A2D	05055	FNHT	M0226	CPT3	FCTF BHF COUNT 0 H
6a	0A2E	05056	FNH1	M0240	CPT3	FCTF FALU CARRY IN H
6b	0A2F	05057	FNHT	M0240	CPT_	FCTF FAMX SEL 0 H

EXPLANATION OF VERSION NUMBERS FOR CONSOLE BOOTING

```
VER P7E-01 MCE-03-10 PDLA-01 CCM PX03-00

I.  WHERE VERSION NUMBERS COME FROM
    A.  PCE (i.e., C in example)
        1.  Version stored in CTD field on location 111 in PCE
    B.  MCB
        1.  Primary version number (i.e., 03 in example)
            a.  Version stored in CID field of location 1111 in
                MCB
        2.  Secondary version number (i.e., 10 in example)
            a.  Stored in CID field of location 111X in MCB
    C.  The rest
        The console puts the microsequencer on the microstack,
        asserts RCP MOP, does a maintenance return; steps the
        microcode to the proper place, and then reads the ID
        address bits (that come from the control store ROM) from
        bits <13:8> of the IB C/B register on CIB. The console
        software then stores the complement of those bits in
        temporaries in IIC) memory.
    D.  PDLA (01 in example)
        The console initiates a maintenance return from location
        "PMC" which is an "RCU" location in the PDLA. The console
        stores the microcode in the location following "PMC" and
        reads the Jump field over the V-Bus, the least
        significant six bits of the Jump field are the version
        number of the PDLA.
    E.  CCM (PX03-00 in example)
        1.  Version of combo + software read from IIIC memory.

II. ERROR MESSAGES
    A.  "Warning MCB & PDLA version mismatch"
        1.  Checks MCB primary version and PDLA versions are the
            same.
    B.  "Fatal-MCE & PCE version mismatch"
        1.  Check MCB secondary version and PCE version is the
            same. Bits <5:2> of the MCB secondary version are
            compared to bits < 3:0> of the PCE version.
```

At any machine check, the error handling microcode attempts to logout the following information. Ordinarily, it appears on the stack as shown, but if a double error halt occurs, the operator can find the same information in the ID Bus temporaries. This information is STAR-specific, of course, and does not apply to other members of the family.

Data	Memory loc'n	ID loc'n	Notes
Byte Count	(SP)	none	40(dec) = 28(hex)
Summary Param	(SP)+4	T0 (30)	See below
CPU Error Status	(SP)+8	T1 (31)	See CES register format
Trapped UPC	(SP)+12	T2 (32)	Microcode error loc'n
VA/VIBA	(SP)+16	T3 (33)	Virtual address
D register	(SP)+20	T4 (34)	
TB ERR 0	(SP)+24	T5 (35)	See TBER0 format
TB ERR 1	(SP)+28	T6 (36)	See TBER1 format
Timeout Addr	(SP)+32	T7 (37)	Physical addr/4
Parity	(SP)+36	T8 (38)	See PARITY format
SBI Error	(SP)+40	T9 (39)	See SBI.ERR format
PC	(SP)+44	none	
PSL	(SP)+48	none	

The summary parameter is a longword. Byte 1 is a flag, which is non-zero if a CP timeout or CP error confirmation interrupt was pending at the time the machine check occurred. The interrupt, if any, has been cleared. Byte zero identifies the type of machine check:

```
80 = CP Read timeout or Error Confirmation Fault
82 = CP Translation Buffer Parity Error Fault
83 = CP Cache Parity Error Fault
85 = CP Read Data Substitute Fault
8A = IB Translation Buffer Parity Error Fault
8C = IB Read Data Substitute Fault
90 = IB Read Timeout or Error Confirmation Fault
9F = IB Cache Parity Error Fault
A1 = Control Store Parity Error Abort
A2 = CP Translation Buffer Parity Error Abort
A3 = CP Cache Parity Error Abort
A4 = CP Read Timeout or Error Confirmation Abort
A5 = CP Read Data Substitute Abort
A6 = Microcode "must supposed to get here" abort
```

"IB" above refers to memory reads generated by the instruction buffer in the process of prefetching the instruction stream. In these cases, the address stored at (SP)+16 is from VIBA. "CP" refers to memory references explicitly requested by microcode and whose address comes from VA.

The CPU will halt if it finds on entry to the error handling microcode that "SPP" is set.

The information on the first error will be in ID[T0-T9] and U-STACK (trapped microaddresses). Unpredictable on CS Parity errors.

The information on the 2nd error will be in the associated error/status registers.

The CPU will be halted after leaving a double error halt code in ID[D.SV].

SUMMARY PARA.	T0
CES	T1
TRAPPED UPC	T2
VA/VIBA	T3
D-REG	T4
TBER0	T5
TBER1	T6
TIME.ADDR	T7
PARITY	T8
SBI.ERR	T9

MICRODIAGNOSTICS RUN, CONSOLE TERMINAL OUTPUT

```
>>>T
        FAILED AT 00013222
>>>T
        IBUF BRD DONE
>>>U

>>>SET                                   ;Console prompt, exit command

ZZ-E5KA6    V5.3                        ;Program title and version
01,02,03,27,                            ;Booting numbers
H0, DF WCS NUMBER = 0003                ;Configuration information
05,06,07,08,09,0A,0B,0C,0D,0E,0F,10,11,12,13,14,15,16,17,
18,19,1A,1B,1C,1D,1E,1F,20,21,22,23,24,25,26,27,28,29,2A,
2C,2D,2E,2F,30,31,32,33,34,35,36,
2HB PASS 0001                           ;Instructions for second half
MOUNT FILE??? ZZ-E5KA6 0 TYPE "U."      ;microdiagnostic monitor prompt
PROMPT
17,1B,39,3A,3B,3C,7D,7E,
CPU TH= 0000007D                        ;Section number(s)
P3780 4K CHIP AT 7A 0L                  ;configuration information
MAX MEMPL= 00090000                     ;Limit is system specific
DA780 AT TR 01
KL781 AT TR 08
RH781 AT TR 28
3F,40,41,42,43,44,45,46,47,48,49,4A,4B,4C,4D,4E,4F,50,
STARTING SBR ADDRS
51,52,53,54,55,56,57,58,59,5A,5B,5C,5D,
END BASE SBR

        DPD FAILED,DCNN TLAA4,STR=NONE,CLOCK HUNG       ;CPU diagnos
        MAP=REX,ADR=0UXS,DAT 1UH0,PTEE=00.EBL-00000000  ;and console
        INIT SBR DONE                                   ;program.
        MAINRD AT 00000000                              ;Refurbis

        [REFURBISH MCS]                                 ;configuratio
        LOAD DONE, 00003000 BYTES LOADED                ;information
        VER: SCS=01 UCB'0F-11 PPLA=0: UCP'003-27 L      ;console prompt
>>>
```

LOAD AND RUN STAND-ALONE MACRODIAGNOSTICS (OFF-LINE)

```
^P                              ; Control P, to return control to the
                                ; console program console I/O mode.
                                ; Make that the 5 position key switch
                                ; on the control panel should be set to
                                ; LOCAL.
                                ; The console program prompt symbol,
                                ; >>>, is displayed.

>>>B<CODE>                      ; <CODE> is the five letter
                                ; mnemonic designation of
                                ; the diagnostic program
                                ; to be loaded via the
                                ; indirect command file.

HALT      ; HALT THE CPU        ; These commands
                                ; and responses are
          CPU HALTED            ; produced automatically
INIT      ; INITIALIZE          ; down to (@EXIT)

          ABIC SEQ DONE
UNJAM     ; AND CLEAR THE BUS

LD XXXXX.XXX/SV:200

          LOAD DONE, 00000000 BYTES LOADED
J         QUICK VERIFY

          LOAD DONE, 00000000 BYTES LOADED
START 1C00

(@DONE)
(@EXIT)

DIAGNOSTIC SUPERVISOR, X2-ESSNA-4.02.417 6-JUN-1979 00:09:21.31

DS>                             ; At this point, any diagnostic
                                ; supervisor command may be typed in,
                                ; in order to modify program execution.

DS> ST                          ; Start the diagnostic program, default
 OC                             ; mode.

DS> ST/SWITCHES                 ; The SWITCHES are optional and refer
                                ; to separately selectable program
                                ; functions or tests.

.                               ; The program starts, identifies
.                               ; itself, and asks the operator for the
.                               ; mnemonic of the device to be tested
.                               ; and for other parameters necessary to
.                               ; program operation.

DS>                             ; Control returns to the diagnostic
                                ; supervisor at the completion of the
                                ; diagnostic program. In order to load
                                ; and run another diagnostic program,
                                ; type Control P and repeat the above
                                ; procedure.
```

NOTE

Operator input is underlined.

LOAD AND RUN MACRODIAGNOSTICS UNDER VMS (ON-LINE)

```
^Y                          ; Return control to VMS.

$RUN ESSAA                  ; Load and run the diagnostic
                            ; supervisor.

                            ; The diagnostic supervisor starts
                            ; up, identifies itself, and prompts
                            ; for input.
```

2. Two sets of commands can be used to load and run a diagnostic program at this point:

```
DS>RL/<CODE>/SWITCHES       ; <CODE> refers to the five letter
                            ; program mnemonic.  The SWITCHES
      (or)                  ; refer to separately selectable
                            ; tests or sections within each
DS>LOAD <CODE>              ; diagnostic program.  These
DS>ST/SWITCHES              ; SWITCHES are program specific.  If
                            ; the SWITCHES are omitted, the pro-
                            ; gram is run in the default mode.
```

MICRODIAGNOSTIC MONITOR COMMANDS

Command/Qlau	Description
DIAGNOST	Initializes the program control flags, and starts microdiagnostic execution at test number one. Valid qualifiers are: /TEST: <NUMBER> -- Dispatch to the test number specified (do not execute any prior tests) and loop on the test indefinitely. /SECTION: <NUMBER> -- Dispatch to the section number specified (do not execute any prior sections) and loop on the section indefinitely. /PASS: <NUMBER> -- Execute the microdiagnostics for specified number of passes before returning to the console. If the number is -1, execute the microdiagnostics indefinitely. /CONTINUE -- This switch is used with the /TEST or /SECT switch to automatically continue after the specified test or section has been reached.

/TEST: <X> <N> -- Dispatch to test <X>, execute tests <N> through <N> (inclusive), and return to command mode.

/TEST: <X> <N> Dispatch to section <X>, execute sections <N> through <N> (inclusive), and return to command mode.

NOTE

In the above two variations of the "/TEST" and "/SECTION" qualifiers, the value of <N> must be less than or equal to <N>. If <N> is less than <N>, testing will start at <N> and continue to the end.

NOTE

/TEST and /SECT cannot be specified simultaneously.

Examples:

DIAG/TEST:22

Dispatch to test number 22 and execute it indefinitely

DIAG/SECT:0

Dispatch to test or number 0 and execute it indefinitely.

DIAG/TEST:-1

Execute all of the micro diagnostics indefinitely.

	DIAG/TEST:2F/CONT Dispatch to test 2F and start execution of the pending tests.
CONTINUE	Continues microdiagnostic execution without changing the program control flags.
Set and Clear Flags	
SET/CLEAR FLAG BD	Sets (or clears) the Halt on Error Detection flag.
S/C/CLEAR FLAG HI	Sets (or clears) the Halt on Error Isolation flag.
SET/CLEAR FLAG LOOP	Sets (or clears) the Loop on Error flag.
set/CLEAR FLAG NER	Sets (or clears) the No Error Report flag.
SET/CLEAR FLAG BELL	Sets (or clears) the Bell on Error flag.
set/CLEAR FLAG ERABT	Sets (or clears) the Error Abort flag.
CLEAR FLAG LS	Clears the Loop on Special Section flag. (Note that this flag cannot be set.)
CLEAR LT FLAG	Clears the Loop on Special Test flag. (Note that this flag cannot be set.)

SET/CLEAR FLAG ALL	Sets (or clears) all of the previous flags.
SET/CLEAR SOMS	Sets (or clears) the Stop on Micro Match bit.
SET/CLEAR SOMM[<ADDRESS>]	Loads address into Microaddress Register and sets (or clears) the stop on Microaddress bit.
SET/CLEAR MM[<ADDRESS>]	Loads <ADDRESS> into the MPA micro mode register.
SET SYSC STATE	Sets the CPU clock to single bus state.
SET STEP BUS	Sets the CPU clock to single bus cycle. Both the SET SYSC STATE and SET STEP BUS commands cause the monitor to enter step mode. Step mode types the current clock state or the UPC value, and waits for terminal input. If a space is typed, the clock is triggered and the current UPC value is typed out. If any other character is entered, step mode is exited.
SET STEP INSTRUCTION	Sets the hardware Single Instruction flag and returns to the monitor. When the hardware traps are invoked, the current value of the Next PC (TPC) is typed. The

	monitor waits for terminal input. If a space is typed, the current pseudo instruction is executed and the current value of the TPC is typed. If any other character is typed, step mode is exited.
SET CLOCK FAST	Sets the CPU clock speed to the fast margin.
SET CLOCK SLOW	Sets the CPU clock speed to the slow margin.
SET CLOCK NOMINAL	Sets the CPU clock speed to normal.
SET CLOCK EXTERNAL	Sets the CPU clock for an external oscillator.
Examine Commands	The following examine commands cause the current microinstruction to be executed before the examine is performed. If it is the first examine since entering the monitor command mode. All successive examines do not execute any additional microinstructions. IO Bus registers T1-T3 are destroyed during the examines, except for the TC Tom and TCom examines. All of the following examines, except V Bus, advance the clock to CPIC before executing the command.

EXAMINE ID <ADDRESS>	Displays the contents of the ID BUS Register specified by <ADDRESS>.
EXAMINE VBUS:<CHANNEL>	Displays the contents of the VBUS channel specified by <CHANNEL>. Bit 0 is at the right side of the display.
EXAMINE RA:<ADDRESS>	Displays the contents of the RA Scratch Pad specified by <ADDRESS>.
EXAMINE RC:<ADDRESS>	Displays the contents of the RC Scratch Pad specified by <ADDRESS>.
EXAMINE LA	Displays the contents of the LA Latch.
EXAMINE LC	Displays the contents of the LC Latch.
EXAMINE DR	Displays the contents of the D Register.
EXAMINE QR	Displays the contents of the Q Register.
EXAMINE SC	Displays the contents of the SC Register.
EXAMINE PE	Displays the contents of the PE Register.
EXAMINE VA	Displays the contents of the VA Register.
EXAMINE PC	Registers the contents of the Program Counter.

Deposit Command — The Deposit command is the same as the examine command, except that the data to be deposited must be supplied by the user.

```
DEPOSIT ID: <ADDRESS> <DATA>
DEPOSIT RA: <ADDRESS> <DATA>
DEPOSIT RC: <ADDRESS> <DATA>
DEPOSIT LA: <DATA>
DEPOSIT LC: <DATA>
DEPOSIT DR: <DATA>
DEPOSIT QR: <DATA>
DEPOSIT SC: <DATA>
DEPOSIT PB: <DATA>
DEPOSIT VA: <DATA>
DEPOSIT PA: <DATA>
```

BLKMTC

BLKMTC <SCR ADDRESS>, [SCR INDEX], <MCS ADDRESS>,
<WORD COUNT>, [<MCS ADDRESS INDEX>]

Move the <WORD COUNT> number of 96-bit microwords from the <SCR ADDRESS>, indexed by <SCR INDEX>, to the MCS starting at <MCS ADDRESS>, indexed by <MCS ADDRESS INDEX>. If an <SCR INDEX> is specified, the <SCR ADDRESS> is indexed by six PDP-11 words (i.e., 96 bits).

If the <MCS ADDRESS> starts with an alpha character, the <MCS ADDRESS> is used as a pointer to a table in the LSI-11 memory. Otherwise, it is used as a physical MCS address.

For example, if the current value of the index is 2, 18g (<SCR INDEX> * 6) would be added to the <SCR ADDRESS> to find the first six-word microword to load into the MCS.

CMPXXT

CMPXXT [<PASS ADDRESS>] , [<FAIL ADDRESS>]

If the error flag, set during a COMPARE instruction (see CMPXXX instructions), is zero, go to the <PASS ADDRESS>. If the error flag is not zero, go to the <FAIL ADDRESS>. If neither a pass or fail address is specified, go to the next instruction in line.

The address of the next instruction is typed. These addresses appear on the typed line named TRACE:.

CLOCK

```
CLOCK <TIMES>
```

Step the system clock <TIMES> number of single time states. If <TIMES> is evenly divisible by four, single bus cycles are executed for each four <TIMES>.

CMPCR

```
CMPCR [<MODE>], <REGISTERS>, <CST ADDRESS>, [<CST ADDRESS
      INDEX>]
```

Compares the contents of the console register specified by <REGISTER> with the contents of the location specified by <CST ADDRESS>, indexed by <CST ADDress index>.

If the <MODE> argument is false, set the error flag. If the <MODE> argument is not specified, it defaults to EQUAL.

If the <REGISTER> argument is specified as IDBMSB or IDBLSB, the register used in the compare is the ID Bus register that was read in the most recent READID instruction.

CMPCAD

CMPCAD [<MODE>], <RBSIECBR>, <DST ADDRESS>, [<DST ADDRESS INDEX>]

Compare by the contents of the console register specified by <REGISTER> and <REGISTER> 2 with the contents of the register specified by <DST ADDRESS> and <DST ADDRESS>+1, indexed by <DST ADDRESS INDEX>.

If the <MODE> argument is false, set the error flag. If the <MODE> argument is not specified, it defaults to EQUAL.

If the <RBSIECBR> argument is specified as TIRDATA or TIRXBUF, the register used in the compare is the IO Bus register that was read in the most recent READIO instruction.

CMPCAM

CMPCAM [<MODE>], <REGISTER>, <MASK ADDRESS>, [<MASK ADDRESS INDEX>], <DST ADDRESS>, [<DST ADDRESS INDEX>]

Take the contents of the console register specified by <RBSIECBR>, mask it with the contents of the <MASK ADDRESS>, indexed by <MASK ADDRESS INDEX>, and compare it with the contents of <DST ADDRESS>, indexed by <DST ADDRESS INDEX>.

If the <MODE> argument is false, set the error flag. If the <MODE> argument is not specified, it defaults to EQUAL.

if the <REGISTER> argument is specified as THROUGH or THROUGHT, the register used in the compare is the TO Bus register that was read in the most recent READ% instruction.

The mask is performed by taking the contents of <MASK ADDRESS>, indexed by <MASK ADDRESS INDEX>, complementing it, and bit clearing the contents of <REGISTER> with it.

EXAMPLE

```
CPPCND [<MODE>], <REGISTER>, <MASK ADDRESS>, [<MASK ADDRESS
        INDEX>], <DIF ADDRESS>, [<DIF ADDRESS INDEX>]
```

take the contents of the constant registers specified by <REGISTER> and <REGISTER+1>, mask it with the contents of <MASK ADDRESS> and <MASK ADDRESS+1>, indexed by <MASK ADDRESS INDEX>, and compare it with the contents of <DIF ADDRESS> and <DIF ADDRESS+1>, indexed by <DIF ADDRESS INDEX>.

If all <REG> arguments is false, set the error flag. If the <MODE> argument is not specified, it defaults to EQUAL.

If the <REGISTER> argument is specified as THROUGH or THROUGHT, the register used in the compare is the TO Bus register that was read in the most recent READ% instruction.

The mask is performed by taking the contents of <MASK ADDRESS> and <MASK ADDRESS+1>, indexed by <MASK ADDRESS INDEX>, complementing it, and bit clearing the contents of <REGISTER> and <REGISTER+1>.

CMPDSV

CMPDSV <DST ADDRESS>, [<DST ADDRESS INDEX>]

Compare the contents of the PC Save register with the contents of the location specified by <DST ADDRESS>, indexed by <DST ADDRESS INDEX>. If the contents are not equal, set the error flag.

ENDLOOP

ENDLOOP <INDEX NAME>

Add the increment value of <INDEX NAME> (see LOOP instruction) to the current value of the index specified by <INDEX NAME>. Compare the current value with the test value (specified in the LOOP instruction). If the current value is less than or equal to the test value, go to the instruction following the most recent LOOP instruction. Otherwise, go to the next sequential instruction.

ERRLOOP

ERRLOOP

Save the address of the next instruction. If an error is detected, and the Loop on Error flag is set (ref. subsection 6.6), execution is restarted at this saved address after the ERRLOOP instruction is executed.

E_CCSB

```
ECCSB <CWCS ADDRESS>, [<WCS ADDRESS INDEX>], [<WCS ROM MODE>]
```

If <WCS ADDRESS> is a numeric string, execute a maintenance return to the location specified by <WCS ADDRESS>, indexed by <WCS ADDRESS INDEX>. If <WCS ADDRESS> is an alpha-numeric string, execute a maintenance return to the location specified by the contents of <WCS ADDRESS>, indexed by <WCS ADDRESS INDEX>. If <ROM MODE> is specified, clear bit 5 of the MSR register during the maintenance return.

E_LOAD

```
ELOAD <DST ADDRESS>, <INDEX NAME>
```

Generate a 16-bit word of all zeros. Insert a logic one in the bit position specified by the current value minus one of <INDEX NAME>, and load this word into the location specified by <DST ADDRESS> and <DST ADDRESS+2>.

E_LOADO

```
ELOADO <DST ADDRESS>, <INDEX NAME>
```

Generate a 32 bit word of all logic ones. Insert a zero in the bit position specified by the current value minus one of <INDEX NAME>, and load this word into the location specified by <DST ADDRESS> and <DST ADDRESS+2>.

IFERROR

IFERROR [<MICROCODE NUMBER>], [<FAIL ADDRESS>]

If the error flag is cleared, type the PC of this instruction, the test number, subtest number, and the good and bad data. Then, go to <FAIL ADDRESS> if the HALTE flag is not set (ref. subsection 4.6).

If the error flag is set, or the <FAIL ADDRESS> is not specified, go to the next instruction.

INITIALIZE

INITIALIZE

Set and clear the CPU initialize bit in the Push on Control register, clear the single Fbus cycle bit, set the single bus cycle bit, and the ADR HOP bit, and set the Freeze bit in the Machine Control register.

PMXGEN

PMXGEN <DEC ADDRESS>, <INDEX FRAME>

Generate the PMUX address specified by the current value minus one of <INDEX FRAME> and load it into the PMUX field of the microinstruction specified by <DEC ADDRESS>.

MICRODIAGNOSTIC PSEUDO INSTRUCTION DEFINITIONS

LDIOREG

LDIOREG <REGISTER>, <SRC ADDRESS>, <SRC ADDRESS INDEX>]

Load the IO Bus register specified by <REGISTER> with the contents of the locations specified by <SRC ADDRESS> and <SRC ADDRESS>+2, indexed by <SRC ADDRESS INDEX>.

If <REGISTER> is the microstack, microcounter, or MCR address, the contents of <SRC ADDRESS> is taken to be 16 bits. Otherwise, it is taken to be 32 bits.

LOADCA

LOADCA <REGISTER>, <SRC ADDRESS>, [<SRC ADDRESS INDEX>

Load the console register specified by <REGISTER> with the contents of the location specified by <SRC ADDRESS>, indexed by <SRC ADDRESS INDEX>. This instruction loads 16 bits of data.

LOOP

LOOP <INDEX NAME>, <START>, <END>, [<STEP DEPENDENT>]

Initialize the loop parameter specified by <INDEX NAME> to the value specified by <START>. Save the value specified by <END> for the ENDLOOP instruction. Calculate and save the increment value for the ENDLOOP instruction via the following algorithm:

> If <START> is less than or equal to <END>, set the increment value to 1; otherwise, set it to -1.

if <END> is in <LABEL NAME>, save the current value of this index name as the <END> value of this index name.

If <ERR ERROR#> is specified, divide the target of <START> and <END> by two if there is only one MCB module on the system. Otherwise, leave them unchanged.

MASK

MASK <KEY ADDRESS>, <MASK ADDRESS>

Take the contents of location <MASK ADDRESS>, complement it, and bit clear the contents of location <DST ADDRESS> with it.

MOVE

MOVE ,<SRC ADDRESS>, [<SRC ADDRESS INDEX>], <DST ADDRESS>

Move the contents of <SRC ADDRESS INDEX> (indexed by <SRC ADDRESS INDEX>) to the location specified by <DST ADDRESS>.

MEMTST

MEMTST <TEST NAME>, [<KEY DESCRIPTION>], [<SCOPE DESCRIPTION>], [<ERROR DESCRIPTION>], [<SYNC POINT DESCRIPTION>]

This instruction creates a test header document for the specified arguments. It clears the error flag, and moves the PC of the next instruction for looping on test.

%READIN

%READIO <REGISTER>

Reads the IO BUS register specified by <REGISTER> and loads the contents of it into locations IOFMSIO and IOREGIO.

%RESET

%RESET

Executes an <ESI-1 reset instruction>.

%REPORT

%REPORT <MODULE NAME STRING>

Types out the module numbers of the modules specified by <MODULE NAME STRING>. If the DAUTI flag is set, return to the Microdiagnostic Monitor.

%SCVB

%SCVB <SRC TABLE ADDRESS>, [<SRC TABLE ADDRESS INDEX>]

Load and read the VBus. Compare the contents of the data at <SRC TABLE ADDRESS>, indexed by <SRC TABLE ADDRESS INDEX>, with the VBus data just read. The <SRC TABLE> has the following format:

```
18:   .WORD    <NUMBER OF BITS TO CHECK>
      VBUSC    <CHANNEL NUMBER>, <BIT NUMBER>, <EXPECTED BIT
               VALUE>

28:   .WORD    <NUMBER OF BITS TO CHECK>
      VBUSC    <CHANNEL NUMBER>, <BIT NUMBER>, <EXPECTED BIT
               VALUE>
```

The following is an example of the <SRC TABLE ADDRESS INDEX>:

```
TSTVB   18,1
```

If the current value of the <SRC TABLE ADDRESS INDEX> is 1, and the <SRC TABLE> looks like the above table, the physical <SRC TABLE ADDRESS> would be 28.

SETPSW

```
SETPSW <DATA>
```

Load the LSI processor status word with the value specified by <DATA>.

SETVEC

```
SETVEC <VECTOR ADDRESS>
```

Set the UNIBUS address specified by <VECTOR ADDRESS> to the expected trap routine.

SKIP

SKIP [<DST ADDRESS>]

Go to the <DST ADDRESS>. If <DST ADDRESS> is not specified, go to the next test. If <DST ADDRESS> starts with the alpha character S, go to the next subtest.

SUBTEST

SUBTEST

Increment the subtest counter.

TYPSIS

TYPSIS

Use the contents of location RADDATA to determine the MOS module configuration and type a message and the number of MOS modules that will be tested. If any of the following conditions exist, the test stream is aborted and the MER (Mo Error Report) flag is set:

a. MOS module count is zero

b. bits 3-0 are nonzero

c. 5-th K-^^ MOS is not present

```
;FIELDS ARRANGED ALPHABETICALLY

ACF/=0,2,7D,D           ;ACCELERATOR CONTROL
        NOP=0
        SYNC=1
        TRAP=2
        CONTROL=3       ;ACCELERATOR-DEPENDENT CONTROL FUNCTION

ACW/=0,3,55             ;ACCELERATOR WISCELLANEOUS CONTROL
        PWR.UP=0
        ABORT=1         ;RETURN ACCEL TO MONITORING IRD
        POLY.DONE=6

ADS/=0,1,AY             ;ADDRESS SELECT
        WR=0
        IOW=1


ALU/=0F,4,66,D          ;ALU CONTROL FUNCTIONS
        A=B=00
        A=B.R.02=01
        A=B+1=02
        INST.DEP=03                     ;INSTRUCTION DEPENDENT
        A+B+1=04
        A+B=05
        A+B.RLD6.D0
        ONAT=07                         ;A .OR. .NOT. B
        XOR=08                          ;A .XOR. B
        ANDNOT=09                       ;A .AND. .NOT. B
        NOTA=0A                         ;.NOT. A
        A=B+PSL.C=0B
        OR=0C                           ;A .OR. B
        AND=0D                          ;A .AND. B
        B=0E
        A=0F


AMX/=0,2,8D             ;AMX TO ALU
        LA=0
        SAMX=1
        RAMX.SXT=2                      ;RAMX SIGN EXTENDED ACCORDING TO DT
        RAMX.DXT=3                      ;RAMX ZERO EXTENDED, DATIL)=0
```

```
BSY/=0,5.72.0           ;BRANCH CHARLIE
        NOP=0                           ;NO BRANCH
        Z=1                             ; ALU Z
        PSR=2                           ;LA<1>, PSL<0>, L64<0>
        C31=3                           ; ALU C31, 0
        ACCEL=4                         ;DECODE FROM ACCELERATOR
        DATA.TYPE=8                     ;(VAX MODE) =, ASRC=>SRC, ASRC=Q+D
                                        ;   0=>LONGW, 1=>QUAD OR DOUBLE
                                        ;   2=>FIELD,  3=>ADDRESS
        END.OP1=A                       ;(I-11 MODE) =, D CLASS, J CLASS+DW27
        IR2=B                           ;(VAX MODE) =, I<3:0>
        PC.MODE=9                       ;(I-11 MODE) =, DW40=SYST+DW40+DW37, DST R=PC
        RET=C4                          ;(VAX MODE) MODE.LSG,ASTLVL, =, =
        SRC.PC=CA                       ;(I-11 MODE) SRC R=PC
        TB.TEST=0B                      ; 0=>TB MISS, 1=>ERROR
                                        ; 2=>STALL, 3=>DATA OK
        MUL=0C                          ;SC.NE.0, D<31>
        SIGNS=0D                        ;Q<31>, D.NE.0, D<31>
        INTERRUPT=0E                    ;NO LOA, INTERNAL INTERRUPT, INT REQ
        DECIMAL=0F                      ;C, 0 BYTE 0 VALID DIGIT, 0F=0 NEG SIGN
        ULTRA=1C                        ;MICROTRAP DISPATCH VECTOR
        LAST.REF=1D                     ;=FPD, NESTED ERROR, LOA TAG BITS:
                                        ; 0=>READ INTERLOCK, 1=>READ, READ ONA
                                        ; 2=>WRITE, 3=>READ, WRITE ONA
        EALU=19                         ;EALU N, FWIN 2, SC.NE).C, SS
        SD=14                           ;SD<9>=RA.NE.0, SD.GT.C, SD<8:5>.NE.0
        ALU1-0-15                       ;ALTG PARITY, ALU<1:0>=0, ALU<1>, ALU<0>
                                        ; (ALU BITS FROM PREVIOUS STATE)
        STATE7-4-16                     ;STATE <7:4>
        STATE3-0-17                     ;DIA E <3:0>
        D.BYTE-18                       ;BYTE 3, 2, 1, 0 OF D.NE.0
        D3-0-1B                         ;D<3:0>
        PSL.CC=1A                       ;N,Z,V,C OF PSL
        ALU=1B                          ;ALU N, ALU Z, IR<0>, ALU C31
        PSL.MODE=1C                     ;(VAX<31:30>, =CONSOLE, USER, KERNEL
        TB.TEST=1D                      ;PTE VALID, ALIGNED, QUAD, =
                                        ; 0=>TRANSLATION OK, 1=>NO CHK AND V=0
                                        ; 2=>ACCESS VIOLATION, 3=>TB MISS
```

```
BMX/=0,3,B2             ;BMX TO ALU
        VMA=0                   ;4 0 IN THE BIT SELECTED BY SC<1:0>
        VMA.DR.LB=1             ;LB UNLESS R-SC, THEN MD
        PACKED.FLOAT=2          ;PACKED FLOATING
        LB=3
        LC=4
        PC=5
        ATX=6
        REMA=7
COK/=0,3,20,0           ;CONDITION CODES
        NC=0                    ;DEFAULT
        LOAD.LSEC=1             ;SAME AS ALU & ALU CONDITIONS
        SH.C=2                  ;SOURCE N, NO EFFECT ON N, Z, C
        TRT.Z=3                 ;LOAD Z IF ALU.NE 0.
                                ;SET N FROM ATX[LOC]
        ROS=4                   ;SET N AND Z FROM ALU[LOT]
        VMA.ALU=5               ;SETNEG UNAFFECTED
        C.ADVSC=6
        TRT.OSP=7
SID/=0,4,42             ;CONSOLE & I/O CONTROL, IR 15/1
        NOW=0                   ;DEFAULT, ALSO ALSO IS STATE
        ACT=5                   ;SET CONSOLE ADVERTISE FLAG
        CONT=7                  ;CLEAR CONSOLE MODE
        READ.SC.9               ;READ IR BUS REG SELECTED BY SC
        READ.ANY.00             ;READ IR BUS REG SELECTED BY JMAX
        WRITE.SC.00             ;WRITE REG SELECTED BY SC
        WRITE.MMAP=01           ;WRITE REG SELECTED BY BMAX

DK/=0,4,B8,C
        NOP=0                   ;DEFAULT, HOLD
        LEFT2=1                 ;DOUBLE SHIFT LEFT
        RIGHT2=2                ;DOUBLE SHIFT RIGHT
        DS=4                    ;IF NOT ALU OP, SHIFT LEFT
                                ;ELSE LOAD FROM SHF
        LEFT=5                  ;SHIFT LEFT
        RIGHT=6                 ;SHIFT RIGHT
        SHF=7                   ;LOAD SHF MUX, INTEGER FORMAT
        SHF.FL=3                ;LOAD SHF MUX, UNPACKED FLOATING FORMAT
        ACCEL=04                ;LOAD ACCELERATOR DATA FROM SHF BUS
        BYTE.SWAP=08            ;REFLECT BYTES AROUND BIT 16
```

```
        Q=0C                            ;LOAD Q THRU SAL
        SAL.SC=03                       ;LOAD SA[SC]
        SAL.SV=05                       ;LOAD SA[SHF VAL]
        CLR=0F                          ;LOAD ZEROS

DT/=0,2,73,0            ;DATA TYPE
                        ;CONTROLS AMX SIGN/ZERO EXTENDER, SHF AMOUNT,
                        ;CONDITION CODE SETTING, AND MEMORY REFERENCES
        LONG=0                          ;DEFAULT
        WORD=1
        BYTE=2
        INSTR.DEP=3                     ;INSTRUCTION DEPENDENT --
                                        ;ANY DI ARDVF, OR QUAD/DOUBLE
EALU/=0,5,19                            ;EXPONENT ALU
        A=0
        OR=1
        ANDNOT=2
        B=3
        A-B=4
        A+B=5
        A+1=6
        NABS(A-B)=7                     ;-ABS(A-B)

SBMX/=0,2,18            ;SBMX TO EALU
        FF=0                            ;DEFAULT
        KMX=1
        AMX.SHF=2
        SHF.VALUE=3                     ;SHIFT VALUE

FPR/=0,1,24,0           ;FP REGISTER CONTROL
        HOP=0                           ;DEFAULT, HOLD
        LOAD=1

FS/=0,1,32              ;FUNCTION SELECT FOR 43-46
        VCT=0                           ;ENABLE MEMORY CONTROL
        CID=1                           ;ENABLE ID BUS AND CONSOLE CONTROL

IER/=0,2,20             ;INTERRUPT AND EXCEPTION ACKNOWLEDGE
        NOP=0
        ISTB=1                          ;STROBE INTERRUPT REQUESTS
        IACK=2                          ;INTERRUPT ACKNOWLEDGE
        EACK=3                          ;EXCEPTION ACKNOWLEDGE
```

```
IOC/=0,4,S2,D           ;IOBUS CONTROL FUNCTIONS
        LOOP=C                  ;LOOPALL
        STOP=1
        FLUSH=2                 ;FLUSH IO BUS  DAC DMA
        START=3
        CLR.0 1=4               ;CLEAR BYTES 0,1 (=11 OPCODE)
        CLR.2.3=5               ;CLEAR BYTES 2,3 (=11 ISTREAM DATA)
        BDEST=7                 ;TRANSFER BRANCH DISPLACEMENT
        CLR.0=08                ;CLEAR BYTE 0 (VAX OPCODE)
        CLR.1=09                ;CLEAR BYTE 1 (VAX SPECIFIER)
        CLR.0-3=0E              ;CLEAR BYTES 0-3 (=11 OP & DATA)
        CLR.1-5.CC=0F           ;CLEAR BYTES 1-5 CONDITIONALLY
                                ; IF THERE IS NO SPECIFIER EVALUATION,
                                ; CLEAR NOTHING.  IF A SELF-CONTAINED
                                ; SPECIFIER, CLEAR AT.  IF INTERMEDIATE,
                                ; RESULTS, OR DISPLACEMENT, CLEAR THE
                                ; ISTREAM LITERAL.

IO.ADDR/=0.5,S6         ;IO BUS ADDRESS
        IBUF=0                  ;RO     ;SPECIFIER/LITERAL DATA FROM AB
        DAY.TIME=1              ;RO=WR  ;DAYSPEC TIME OF DAY...
                                ; MUST READ UNTIL STOPS CHANGING
        SYS.ID=3                ;RO     ;SYSTEM IDENTIFICATION
        RXCS=4                  ;RO=WR  ;CONSOLE RECEIVE CONTROL/STATUS
        RXDB=5                  ;RO     ;CONSOLE RECEIVE DATA BUFFER
                                ; (FROM ID REGISTER)
        TXCS=6                  ;RO=WR  ;CONSOLE TRANSMIT CONTROL/STATUS
        TXDB=7                  ;WO     ;CONSOLE TRANSMIT DATA BUFFER
                                ; (FROM ID REGISTER)
        CD=8                    ;DATA PATH D/O REGISTERS (MAINT ONLY)
        NXT.PER=9               ;WO     ;INTERVAL CLOCK NEXT PERIOD REGISTER
        CLK.CS=0A               ;RO=WR  ;INTERVAL CLOCK CONTROL/STATUS
        INTERVAL.CNT=0B         ;RO     ;CURRENT INTERVAL COUNT
        CES=0C                  ;RO=WR  ;CACHE ERROR/STATUS
        VECTOR=0D               ;RO=WR  ;EXCEPTION & INTERRUPT CONTROL
        SCR=0E                  ;RO=WR  ;SOFTWARE INTERRUPT REGISTER
        PSL=0F                  ;RO=WR  ;PROCESSOR STATUS LONGWORD
        TBUF=10                 ;TRANSLATION BUFFER DATA
        TBER0=11                ;TB ERROR/STATUS 0
        TBER1=12                ;TB ERROR/STATUS 1
        ACC.D=13                ;ACCELERATOR REGISTER =0
```

MICRODIAGNOSTICS CONTROL ROM FIELD DEFINITIONS

```
        ACC.1=15                        ;ACCELERATOR REGISTER #1
        ACC.2=16                        ;ACCELERATOR REGISTER #2
        ACC.CS=17                       ;ACCELERATOR CONTROL/STATUS
        SILE=18                         ;NEXT ITEM FROM SBI HISTORY
        SBI.ERR=19                      ;SBI ERROR REGISTER
        TIMEOUT.ADDR=1A                 ;SBI TIMEOUT ADDRESS
        FAULT=1B                        ;FAULT STATUS
        COMP=1C                         ;SBI SILO COMPARATOR
        MAINT=1D                        ;SBI MAINTENANCE
        PARITY=1E                       ;CACHE PARITY
        USTACK=20                       ;MICROSTACK
        UBREAK=21                       ;MICRO BREAK
        WCS.ADDR=22
        WCS.DATA=23             ;WS     ;WRITING WCS POINTS ADDRESS
;IO BUS ADDRESSES CONTINUED.    ADDRESSES THAT ARE RAM LOCATIONS

        P0BR=24                         ;PROCESS SPACE 0 BASE REGISTER
        P1BR=25                         ;PROCESS SPACE 1 BASE REGISTER
        SBR=26                          ;SYSTEM SPACE BASE REGISTER
        KSP=27                          ;KERNEL STACK POINTER
        ESP=29                          ;EXEC STACK POINTER
        SSP=2A                          ;SUPERVISOR STACK POINTER
        USP=2B                          ;USER STACK POINTER
        ISP=2C                          ;INTERRUPT STACK POINTER
        FPDA=2D
        DUMMY=2E
        DUMMY=2F
        T0=30                           ;GENERAL TEMPS
        T1=31
        T2=32
        T3=33
        T4=34
        T5=35
        T6=36
        T7=37
        T8=38
        T9=39
        PCBB=3A                         ;PROGRAM CONTROL BLOCK BASE
        SCBB=3B                         ;SYSTEM CONTROL BLOCK BASE
        P0LR=3C                         ;PROCESS 0 LENGTH REGISTER
        P1LR=3D                         ;PROCESS 1 LENGTH REGISTER
        SLR=3E                          ;SYSTEM LENGTH REGISTER
```

```
U:=0,13,0,1             ;NEXT MICRO WORD ADDRESS. DEFAULT IS THE
                        ;FOLLOWING MICRO WORD
                        ;SYMBOLS ARE DEFINED BY ":"
KMB/KQ,6.59             ;CONSTANTS OR 1 SHIFT FK
        .0=0                        ;=0 FROM FK
        .1=1                        ;=1 FROM FK
        .2=2                        ;=2 FROM FK
        .3=3                        ;=3 FROM FK
        .4=4                        ;=4 FROM FK
        SP1.CON=5                   ;SPECIFIER 1 CONSTANT
        SP2.CON=6                   ;SECIFIER 2 CONSTANT (-1) MODE)
        ZERO=8                      ; ON ZEROS (VAX MODE;
        SC=7                        ;SC[5:0] FROM PA
                        ;6 - SP1 CONSTANTS IF CYCLE SETUP IF ALU IN ARITH MODE)
                                ;DECIMAL VALUE OF CONSTANT
        .14=8                       ;20     (AF,JL,NH)
        .40=9                       ;160    (AF,JL)
        .34=34                      ;92     (AF)
        .28=28                      ;40     (AF)
        .40=35                      ;64     (AF,JL,NH,TF)
        .50=28                      ;80     (AF,VH)
        .3000=2F                    ;12000  (JL)
        .DF=DF                      ;235    (JL)
        .80=10                      ;128    (AF,JL,NH,TF)
        .8000=11                    ;-32768 (AF)
        .FF=12                      ;255    (VH,TF)
        .FFC0=13                    ;-256   (VH,AF,JL)
        .1E=14                      ;30     (AF)
        .9F=15                      ;159    (VH,AF,TF)
        .7F=16                      ;127
        .F=17                       ;7      (AF,VH)
        .F=18                       ;15     (VH,EM,AF,TF)
        .10=19                      ;16     (VH,AF,JL,TF)
        .FFE9=1A                    ;-24    (NH,TF)
        .FFF0=1B                    ;-16    (CN,JL,TF,VH)
        .FFF8=1C                    ;-8     (CN,TF,VH)
        .20=1D                      ;32     (CN,JL,VH,TF)
        .3C=1E                      ;48     (NH,AF,VH,TF)
        .18=1F                      ;24     (NH,AF,TF)
        .8FF=20                     ;1023   (CN)
```

```
          .C=2'                         ;13     (CN,GL,TF,WH)
          .D=30                         ;13     (TF)
          .1F=93                        ;31     (AF,GL,WH,TF)
          .1F00=94                      ;7936   (GL,WH)
          .80=35                        ;176    (WH)
          .E000=36                      ;       (CN)
          .7C=37                        ;124    (AF)
          .FFEQ=38                      ;-52    (LL)
          .50=39                        ;86     (TF)
;         SPARE=3A
          .3F01=26                      ;9      (LL)
          .FFEF=3C                      ;-17    (AF)
          .FFF1=3D                      ;-15    (AF)
          .19.3E                        ;25     (AF)
          .FFF9=3F                      ;-7     (AF)
; 4WA DEFINITION CONTINUED

          .FFFF=30                      ;-1     (WH,GL,TF)
          .38=31                        ;135    (AF)
          .3020=32                      ;7      (TF)
          .F0=33                        ;240    (TF)
          .C0=34                        ;192    (TF,WH)
          .6=35                         ;6      (CY,GL,TF)
          .8=36                         ;8      (CY)
          .FFF0=37                      ;-10    (CY)
          .FFF5=38                      ;-11    (CY)
          .14=39                        ;38     (CY,AF,TF)
          .24=3A                        ;76     (CY,WH)
          .1E=3B                        ;37     (CY,AF,TF)
          .FFC2=3C                      ;-4     (CY,LL,WH)
          .A=3D                         ;10     (AF,WH)
          .7F=3F                        ;126    (AF,TF)
;         SPARE=3F
MCT/-03,6,42,0                  ;MEMORY CONTROL
          TEST.RCHK=00                  ;TEST TRAP WITH READ CHECK
          NCY.NOP=02                    ;NEITHER CPU NOR IO GETS MEM CYCLE
          TEST.WCHK=04                  ;TEST TRAP WITH WRITE CHECK
          WRITE.V.NCCHK=0A              ;WRITE, INHIBIT TRAPS
          WRITE.V.WCHK=0C               ;WRITE, NORMAL VARIETY
          LOCKWRITE.V.WCHK=0E           ;INTERLOCK WRITE, VIRTUAL ADDRESS
```

```
        READ.V.NORM=10                  ;READ, NORMAL VARIETY
        READ.V.NOINH=12                 ;READ, INHIBIT TRAPS
        READ.V.VCHK=14                  ;READ FOR VCHK=
        READ.V.IBUFF=15                 ;READ, DATA CONTINUED BY IBUFF=R
        READ.V.IFET=18                  ;BEGIN NEW INSTRUCTION STREAM
                                        ; DATA GOES TO INSTRUCTION BUFFER

        LOCKREAD.V.NOINH=1A             ;INTERLOCK READ, INHIBIT CHECK
        LOCKREAD.V.NORM=1C              ;INTERLOCK READ, NORMAL VARIETY
        SBI.HOLD=20                     ;STOP ALL SBI ACTIVITY
        SBI.HOLD=-UNJAM-22              ;RESET SBI
        INVALIDATE=24                   ;CLEAR CACHE ENTRIES
        VALIDATE=26                     ;MICRODIAGNOSTIC FORCE VALID
        EXTWRITE.P=28                   ;EXTENDED WRITE TO CLEAR MOS ERRORS
        WRITE.P=2A                      ;WRITE, PHYSICAL
        LOCKWRITE.P=2C                  ;INTERLOCK WRITE, PHYSICAL
        READ.P=2E                       ;READ, PHYSICAL
        READ.INT.ULN=30                 ;INTERRUPT BUNTARY READ
        LOCKREAD.P=32                   ;INTERLOCK READ, PHYSICAL
        ALLOW.IS-READ=3E                ;GIVE IB A CYCLE IF IT WANTS ONE

MSC/=0,4,28,0
        NOP=0                           ;DEFAULT
        CHK.CHM=01                      ;CREATE NEW PSL FOR CHM
        CHK.FLT.IPR=02                  ;DISAP IF ALUKISP=1, ALUCKIP=0
        CHK.USE.ADDR=03
        IRD=04                          ;THIS STATE IS INSTRUCTION DECODE
        LOAD.STATE.05
        LOAD ACC.CC=06                  ;TAKE CONDITION CODES FROM MODE.REGISTER
        READ.R.OS=07                    ;READ R[SP BLOCK STACK]
        CLR.FPD=08                      ;CLEAR PSL<FPD> BIT
        SET.FPD=09                      ;SET DATA
        CLR.NEST.ERR=0A                 ;CLR NESTED ERROR FLAG IN CPU STATUS
        SET.NEST.ERR=0B                 ;SET DATA
        SECOND.REF=0C                   ;OP UNALIGNED DATA REFERENCE
        RETRY.NO.TRAP=0D                ;APPLY SAVED CONTEXT, INHIBIT TRAPS
        RTTRY.TRAP.0F                   ;APPLY SAVED CONTEXT TO THIS REF
        INIT.ON.ADDR.0F                 ;ALLOW USE OF FULL 32-BIT ADDRESS

PCM/=0,3,32,0           ;ADDRESS COUNT CONTROL
        NOP=0                           ;DEFAULT
```

```
        PC_VA:1
        PC_IB:2
        VA=0:3                          ;VA_VA+1
        PC:1=4                          ;PC_PC+1
        PC+2=5                          ;PC_PC+2
        PC+4=6                          ;PC_PC+4
        PC=N:7                          ;PC_PC+N,  N IS DETERMINED BY INST'S BUFFER

OK/=0,4,5 :3
        NOP=0                           ;DEFAULT, HOLD
        LEFT2:1                         ;DOUBLE SHIFT LEFT 2
        RIGHT2:2                        ;DOUBLE SHIFT RIGHT 2
        LEFT:3
        RIGHT:6
        SHF:8                           ;LOAD S+F, INTEGER FORMAT
        SHF.-L:9                        ;LOAD S++, UNPACKED FLOATING FORMAT
        DSC CON=0A                      ;DECIMAL CONSTANT = N+1 IN EACH NIBBLE
                                        ; FOR WHICH ALL DRY BIT IS FALSE
        ACCF:0B                         ;LOAD ACCELERATOR DATA FROM OF BUS
        D:0C
        TD:0E                           ;LOAD TD BUS
        CLR:0F                          ;LOAD ZERO

RAMX/=0,1,77,0          ;DATA PATH MUXER TO ANX
        D=0                             ;DEFAULT
        D=1
RMMX/=0,1,77            ;DATA PATH MUXER TO RMX.  SAME BIT AS RAMX
        C:0
        D=1
SCR/=0,1,33,0           ;SC REGISTER CONTROL
        NOP=0                           ;DEFAULT, HOLD
        LOAD=1                          ; DOE SCR(09:00)

SGN/=0,3,43,0           ;SIGN CONTROLS
        NOP=0                           ;DEFAULT
        LOAD.SS=1                       ;SS_ALU<15>
        SS.FROM.SD=2                    ;SS_SD
        NOT.SD=3                        ;SD_NOT SD
        SD.FROM.SS=4                    ;SD_SS
        SS.XOR.ALU=5                    ;SU_ALU<15>, SS_SS.XOR.ALU<15>
        SD0.SU=6                        ;SD_ALU<15>, SS_SS.XOR.ALU<15>.XOR.IR<1>
        CLR.SD:SS=7                     ;CLEAR BOTH
```

```
SHF/40,8.BF.D           ;ALU SHIFTER CONTROLS
        ALL=0                       ;SHF_ALU, SHF_ALU
        LEFT=1                      ;SHF_ALU[31:1], INSERT SI CNTL
        RIGHT=2                     ;SHF_ALU[31], INSERT SI CNTL
        ALL.BIT=3                   ;SHF_ALU[DST (3,11,19,27)], INSERT 0
        RIGHT3=4                    ;SHF_ALU[RC0], INSERT SI CNTL
        LEFT3=5                     ;SHF_ALU[31]

SI/=8,8.SS.D            ;SHIFT INPUT CONTROLS
                                ;       SEL     0       9
                                ;       ---     -       -
        DIVD=0                  ;       PSL<N>  Q31     ALU Q31
        ASHR=1                  ;       ALU 31  Q0      Q9
        ASHL=2                  ;       0       0       Q8
        ZERO=3                  ;       0       0       9
        SPARE=4
        DI=5                    ;       Q31     Q9      ALU Q31
        VAL=6                   ;       0       ALL 0,1 0
        VAL=7                   ;       1       ALL 0,1 1

SVX/=0,2,.S             ;MIXER TO SC
        EALU=0                      ;EALL <9:0>
        PE=1                        ;-E<9:0>
        ALU=2                       ;ALU<09:00>
        ALU.EAP=3                   ;ALU<19:09>

SPD/=0,7,BS,D                   ;SCRATCH PAD OPCODE, 7 BITS
        NOP=0                       ;DEFAULT
        LOAD.LC.RC=5                ;LOAD LC, ADR=SC[09:00]
        WRITE.RC.SC=7               ;WRITE RC, ADR=SC[09:00]

SPD.AC/=0,4.38          ;4 FUNCTION BITS OF SPD FIELD
        LOAD.LA=1                   ;LOAD LA, LB FROM R[ACN]
        LOAD.LA=2                   ;LOAD LA,RN, HOLD LB
        WRITE.RAB=3                 ;WRITE R4, RB (ACN)

SPD.ADR/=0.5,BS         ;AC NUMBER IN SPD FIELD
                ;VAX MODE       RA              RB
        SP1.SP1=0               ;0      SP1 R           SP1 R
        SP2.SP2=1               ;1      SP2 R           SP2 R
```

```
        SP2,SP1=2                       ;2      SP2 R           SP1 R
        PRN=3                           ;3      PRN             PRN
        PRN+1=4                         ;4      PRN+1           PRN+1
        SC=5                            ;5      SC<03:00>       SC<03:00>
        SP1+1=6                         ;6      SP1 R+1         SP1 R+1

SPD.ADR[1/=6,3,35       ;AC NUMBER IN SPD FIELD == 11 MODE
                        ;*** MODE       R6              R5
        SRC.SRC=0                       ;0      SRC R           SRC R
        DST.DST=1                       ;1      DST R           DST R
        DST.SRC=2                       ;2      DST R           SRC R
        ;SRC,SRC=3                      ;3      SRC R           SRC R
        SRC.OR.1=4                      ;4      SRC R .OR. 1    SRC R .OR. 1
        SC=5                            ;5      SC<03:00>       SC<03:00>

SPD.R/=0,3,32           ;SCRATCH PAD FUNCS WITH LOW 4 BITS OF SP AS ADR
        LOAD.LC=3                       ;LOAD LC, ADR=SPD.RN
        WRITE.RC=3                      ;WRITE RC
        LOAD.LAR=4                      ;LOAD LA, LB
        WRITE.RAB=5                     ;WRITE RA, RB
        LOAD.LAB!.WRITE.RC=6            ;LOAD LA, LB[R1], AND WRITE RC[RN]
        LOAD.LC!.WRITE.RAB!=7           ;LOAD LC[RN], AND WRITE RA, RB[R1]
SPD.RAB/=0,3,35         ;RA/RB LOCATIONS
        R0=0
        R1=1
        R2=2
        R3=3
        R4=4
        R5=5
        R6=6
        R7=7
        AP=0C                   ;R12 = ARGUMENT LIST POINTER
        FP=0D                   ;R13 = STACK FRAME POINTER
        SP=0E                   ;R 14 = STACK POINTER
        R15,0F                  ;R15 = PC, TO SOMEWHERE, SCRATCH TO USCDE
SPD.RC/=0,4,35          ;RC LOCATIONS
         0=0
         1=1
        2=2
        3=3
```

```
        TC=4
        TS=5
        TS=6
        TT=7
        LD.SV=F                         ;NEW VLMT SAVES LC HERE
        VA.SV=8
        VTE.VA=0A
        PTF.PA=0B
        PC.SV=0C
        TC.SV=0D
        VA.RET=0E
        MBLT.VA=0F
        PTE.NA=0F

SUB/=0,3,64.D           ;SUBROUTINE CONTROL
        NOP=0                           ;DEFAULT
        CALL=1                          ;PUSH CPU OF THIS MICRODIAGNSTRUCTION
                                        ; RETURN ADDR
        RET=2                           ; OP   OF MICRODIAGNOSTIC TO JMP
                                        ; AND POP RETURN ADDR
        SPEC=3                          ;RESERVED (ON R DAYS OF MDX)
                                        ; (PC WITH SPECIFIER STORED FROM
                                        ; INSTRUCTION BUFFER

VAR/=0,1,25.D
        MD=0                            ;DEFAULT
        LOAD=1                          ;LOAD VA


;ALU_D. . THRU ALU_E.AND...

ALU_D[A]        'ANX/RAVX.BXT.DT/ ONE.ALU/A'
ALU_D[]D        'ALU/B1.ANX/RAMX.BXT.LONG.BMA/D=VX.BMA/D'
ALU_D=D         'ANX/RAVX.BXT.DT/LONG.RMX/D.BMA/BMVA.ALU/A=D'
ALU_D=D-1       'ANX/RAMX.BXT.DT/LONG.RMX/D.BMA/BMVA.ALU/A=D-1'
ALU_D=D         'ANX/RAMX.BXT.DT/LONG.RMX/D.BMA/BMVA.ALU/A=D'
ALU_D=K[]       'ANX/RAMX.BXT.DT/LONG.RMX/#1.BMA/RMX.ALU/A=D'
ALU_D=K[]+1     'ANX/D' ANX/RMX.ANX/RAMX.BXT.DT/ ONE.ALU/A=D+1'
ALU_D=K[]       'ANX/D' ANX/RMX.ANX/RAMX.BXT.DT/ ONE.ALU/A=D'
ALU_D=K[]+1     'ANX/D' ANX/RMX.ANX/RAMX.BXT.DT/LONG.ALU/A=D+1'
```

```
ALU_Q+LC-1          "AVX/RAMX.CXT,DT/LONG,BMX/LC,ALU/A+B-1"
ALU_Q+LC            "AVX/RAMX.CXT,DT/LONG,BMX/LC,ALU/A+B"
ALU_Q-LC            "AVX/RAMX.CXT,DT/LONG,BMX/LC,ALU/A-B"
ALU_Q+LC+1          "AVX/RAMX.CXT,DT/LONG,BMX/LC,ALU/A+B+1"
ALU_Q-LC-1          "AVX/RAMX.CXT,DT/LONG,BMX/LC,ALU/A-B-1"
ALU_Q+MASK+1        "AVX/RAMX.CXT,DT/LONG,BMX/MASK,ALU/A+B+1"
ALU_D+Q             "AVX/RAMX.CXT,DT/LONG,RBVX/D,BMX/RBVX,ALU/A+B"
ALU_D-Q             "AVX/RAMX.CXT,DT/LONG,RBVX/D,BMX/RBVX,ALU/A-B"
ALU_D-Q-1           "AVX/RAMX.CXT,DT/LONG,RBVX/D,BMX/RBVX,ALU/A-B-1"
ALU_D+Q+1           "AVX/RAMX.CXT,DT/LONG,RBVX/D,BMX/RBVX,ALU/A+B+1"
ALU_-1              "AVX/RAMX.CXT,DT/LONG,ALU/MINUS1"
ALU_D               "RAMX/D,AVX/RAVX,ALU/A"
ALU_D.CXT[]         "RAMX/D,AVX/RAVX.CXT,DT/@1,ALU/A"
ALU_D.CXT[].AND.K[]         "RAVX/D,AVX/RAMX.CXT,DT/@1,KMX/@2,BMX/KMX,ALU/AND"
ALU_D.CXT[].ANDNOT.K[]      "ALU/ANDNOT,AVX/RAMX.CXT,DT/@1,RAMX/D,BVX/KMX,KMX/@2"
ALU_D.CXT[]+K[]     "RAMX/D,AMX/RAVX.CXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_D.CXT[]-K[]     "RAMX/D,AMX/RAVX.CXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B"
ALU_D.CXT[]+LC      "ALU/A+B,AMX/RAMX.CXT,DT/@1,RAVX/D,BMX/LC"
ALU_D.CXT[]!Q       "ALU/A!B,AMX/RAMX.CXT,DT/@1,RAVX/D,BMX/RBMX,RBMX/Q"
ALU_D.CXT[]-Q       "RAMX/D,AMX/RAVX.CXT,DT/@1,RBMX/Q,BMX/RBVX,ALU/A-B"
ALU_D.AND.K[]       "RAVX/D,AMX/RAVX,KMX/@1,BMX/KMX,ALU/AND"
ALU_D.AND.MASK      "RAVX/D,AMX/RAVX,BMX/MASK,ALU/AND"
ALU_D.ANDNOT.K[]            "RAMX/D,AMX/RAVX,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_D.ANDNOT.MASK           "RAMX/D,AMX/RAVX,BMX/MASK,ALU/ANDNOT"
ALU_D.ANDNOT.Q      "RAVX/D,AMX/RAVX,RBMX/Q,BMX/RBVX,ALU/ANDNOT"
;ALU_D(R)... THRU ALU_D.XOR...

ALU_D!(Q)           "RBVX/D,BMX/RBMX,ALU/B"
ALU_D!K[]           "RAVX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/A!B"
ALU_D+K[]           "RAVX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B"
ALU_D-K[]           "RAVX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B"
ALU_D-K[]+1         "RAVX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B+1"
ALU_D-K[]-1         "RAVX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1"
ALU_D-[illegible]   "RAVX/D,AMX/RAMX,BMX/LE,ALU/A-C"
ALU_D!LC            "RAVX/D,AMX/RAMX,BMX/LC,ALU/A!B"
ALU_D+LC            "RAVX/D,AMX/RAMX,BMX/LC,ALU/A+B"
ALU_D-LC            "RAVX/D,AMX/RAMX,BMX/LC,ALU/A-B"
ALU_D-LC-1          "RAVX/D,AMX/RAMX,BMX/LC,ALU/A-B-1"
ALU_D+LC+PSL.C      "RAVX/D,AMX/RAMX,BMX/LC,ALU/A+B+PSL.C"
ALU_D.OR.K[]        "RAVX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR"
ALU_D.OR.LC         "RAVX/D,AMX/RAMX,BMX/LC,ALU/OR"
```

```
ALU_3.ORNOT.MASK        RAMX/J,RAM/RAMA,SMX/MASK,ALU/ORNOT*
ALU_3.OR.O              *SMX/D,RAM/RAMA,SMX/O,EMA/MSK,ALU/OR*
ALU_0,J2                *SMX/D,RAM/RAMX,REMX/Q,EMA/MSMX,ALU/R1*
ALU_0+Q                 *SMX/D,RAM/RAMX,REMX/Q,EMA/MSMX,ALU/A+Q*
ALU_J-Q                 'SMX/D,RAM/RAMX,REMX/Q,B'S'/R'SMX,ALU/A-Q*
ALU_J+Q-1               *SMX/D,RAM/RAMX,REMX/Q,B'S/R'SMX,ALU/A+Q-1*
ALU_J-Q-1               *SMX/D,RAM/RAMX,REMX/Q,B'S/R'SMX,ALU/A-Q-1*
ALU_J+0+PSL.C           *SMX/D,RAM/RAMX,REMX/Q,EMA/MSMX,ALU/A+0+PSL*
ALU_J.SXT[]             *SMX/D,RAM/RAMX,SXT,EMA/MSMX,ALU/A*
ALU_J.SXT[].#NJ.R[]     *SMX/D,RAM/RAMX,SXT,EMA/MSMX,ALU/A*
ALU_J.SXT[]#R[]         *SMX/D,RAM/RAMX,SXT,DT/R1,RMX,'S,RMX/RMX,ALU/A+R*
ALU_J.RMR.R[]           *SMX/D,RAM/RAMX,RMR/R,BUS/RMX,ALU/RMR*
ALU_J.RMR.LO            *SMX/D,RAM/RAMX,RMR/C,ALU/RMR*
ALU_J.XOR.Q             *SMX/D,RAM/RAMX,REMX/Q,EMA/MSMX,ALU/XOR*
ALU_J.XOR.R[]           *SMX/D,RAM/RAMX,REMX/Q,EMA/MSMX,ALU/XOR*
ALU_J.XOR.RSC[]         *SMX/D,RAM/RAMX,SRO,R/LO3...C,SRO,RC/R1,RMX/.C,ALU/XOR*
;ALU_K. . THRU ALU_PL...

ALU_K[]                 *RMX/R1,EMX/RMA,ALL/B*
ALU_LA                  *RMX/14,ALU/A*
ALU_LA.RMX.K[]          *RMX/14,EMX/R1,SMX/RMX,ALU/A*
ALU_LA.RMX#2-.K[]       'RMX/LA,RMX/R1,SMX/RMX,ALU/A+B'
ALU_LA.RMX#2-.MASK      'RMX/LA,RMX/R1,SMX/RMX,ALU/ANDNOT'
ALU_LA.XOR.LO           *RMX/LA,EMX/C,ALU/XOR*
ALU_LA.0                *RMX/L4,RMX/R1,SMX/EMA,ALU/R1*
ALU_LA+C                *RMX/L4,RMX/R1,SMX/EMX,ALU/A+B*
ALU_LA+C-1              *RMX/L4,RMX/R1,SMX/EMX,ALU/A+B-1*
ALU_LA+K[]              *RMX/14,RMX/R1,SMX/EMX,ALU/A+B*
ALU_LA-K[]              *RMX/14,RMX/R1,SMX/EMX,ALU/A-B*
ALU_LA+K[]+1            *RMX/14,RMX/R1,SMX/EMX,ALU/A+B+1*
ALU_LA+K[].#LOG         'RMX/14,RMX/R1,SMX/EMX,ALU/A+B.#LOG'
ALU_LA+K[].#LOG         'RMX/L4,RMX/R1,SMX/EMX,ALU/A+B.#LOG'
ALU_LA+LB               *RMX/LA,SMX/LB,ALU/A+*
ALU_LA+LB               ALU.#LOG/LB,ZHB/-2*
ALU_LA[]0               'RMX/14,RMX/R1,SMX/EMX,ALU/R1'
ALU_LA[]Q               'ALU.#+B,RMX/LA,EMA/MSMX,REMX/Q'
ALU_L4+Q                'ALU.#+B,RMX/LA,EMA/MSMX,REMX/Q'
ALU_L4-Q                'ALU.#-B,RMX/LA,EMA/MSMX,REMX/Q'
ALU_L4-Q-1              'ALU.#-B-1,RMX/LA,EMA/MSMX,REMX/Q'
ALU_LB                  *SMX/LB,ALU/B*
ALU_LC                  *SMX/C,ALU/B*
```

```
ALU_NOT.0       "ALU/NOT4,ANX/RAYX,RAYX/0"
ALU_NOT.RC[]    "RPO.R/LOAD.LC.RPO.RO/0",BMX/.C.AMX/RANX.DXT,DT/LOAD,ALU/DRnDT"
ALU_PHC4,PP     "BYX/PACKED.F..ALU/8"
ALU_PC          "BYX/PC,ALU/8"
:ALU_0... THRU CACHE_...

ALU_0           "RAMX/Q.AYX/RANX,ALU/A"
ALU_0.DXT[]     "RAMX/Q.AYX/RANX.DXT,DT/P1,ALU/A"
ALU_0.DXT[].ANDNOT.R[]  "ALU/ANDNOT.AYX/RAMX.DXT,DT/$",RANX/Q,BMX/RNX,RNX/P2"
ALU_0.DXT[].CF,R[]      "ALU/CF,ANX/RAYX.DXT,DT/P1",RAMX/Q,BMX/RMX,RNX/P2"
ALU_0.DXT[].CF.C        "ALU/CF,AMX/RAYX.DXT,DT/P1",RAMX/Q,BMX/RBMX,RBMX/D"
ALU_0.DXT[]+B   "A.L/A+1,AMX/RAYX.DXT,DT/P1,BYX/RBMX,RBYX/D,RAYX/Q"
ALU_0.DXT[]+C+1 "ALU/A+B+1,AYX/RAMX.DXT,DT/@1,BNX/RBYX,RANX/Q,RBNX/D"
ALU_0.DXT[]+K[] "ALU/A+B,AMX/RAYX.DXT,DT/P1,RAMX/Q,BYX/RMX,RMX/P2"
ALU_0.DXT[]-K[] "ALU/A-B,AMX/RAYX.DXT,DT/P1,RAMX/Q,BYX/RMX,RMX/P2"
ALU_0.AND.K[]   "RAMX/Q,AYX/RAMX,RNX/P1,BYX/AYX,ALU/AND"
ALU_0.ANDNOT.MASK       "RANX/Q,AMX/RAYX,BYX/MASK,ALU/ANDNOT"
ALU_0.ANDNOT.K[]        "RANX/Q,AMX/RAYX,AYX/P1,BYX/RMX,ALU/ANDNOT"
ALU_0[]$1       "RBMX/Q,AYX/RBMX,ALU/0
ALU_0[]0        "RAMX/Q,AYX/RAMX,RBYX/0,TNX/RBYX,ALU/$1"
ALU_0-0         "RAMX/Q,AYX/RAMX,RBYX/0,SNX/RBYX,ALU/A-B"
ALU_0-0/1       "A.L/A-B-1,ANX/RAMX,RAYX/Q,SNX/RBYX,RBMX/D"
ALU_0+P[]       "RAMX/Q,AYX/RAMX,RMX/P1,SNX/ANX,ALU/A+B"
ALU_0-K[]       "RAMX/Q,AYX/RAMX,RMX/P1,SNX/ANX,ALU/A-B"
ALU_0+A[]+1     "ALU/A+B+1,ANX/RAMX,RAYX/Q,SNX/RNX,RNX/P1"
ALU_0+LB        "RAMX/Q,AYX/RAMX,BNX/LB,ALU/A+B"
ALU_0-LB        "RAMX/Q,AYX/RAMX,BMX/LB,ALU/A-B"
ALU_0+LB+1      "RAMX/Q,AMX/RAMX,BNX/LB,ALU/A+B+1"
ALU_0+.C        "RAMX/Q,AYX/RAMX,BNX/LC,ALU/A+B"
ALU_0-.C        "RAMX/Q,AYX/RAMX,BMX/LC,ALU A-B"
ALU_0+.C+1      "ALU/A+B+1,ANX/RAMX,RAYX/Q,BMX/.C"
ALU_0+MASK      "ALU/A+B,AYX/RANX,RAMX/Q,BYX/MASK"
ALU_0-MASK-1    "ALU/A-B-1,ANX/RAMX,RANX/Q,BMX/MASK"
ALU_0.OR.K[]    "RAYX/Q,ANX/RAYX,BMX/P1,BNX/ANX,ALU/OR"
ALU_0.OR.LC     "RAYX/Q,ANX/RAMX,BMX/LC,ALU/OR"
ALU_0.ORNOT.K[] "ALU/ORNOT,AMX/.AMX.RAYX/Q,BMX/BNX,RNX/P1"
ALU_0.SAT[]     "ALU/A,AMX/RAMX.SX.C.'/2',RAMX/Q"
ALU_0.SAT[].ANDNOT.K[]  "ALU/ANDNOT,AMX/RAMX.SXT,RANX/Q,BMX/RMX,RMX/P2,D1/P1"
ALU_0.SAT[]-A[] "RAYX/Q,ANX/RAYX.SX",DT/P1,RNX/P2,BNX/RNX,ALU/A+B"
ALU_0.SAT[]-LB  "RAYX/Q,ANX/RAMX.SX",DT/P1,BMX/.B,ALU/A+B"
```

MICRODIAGNOSTICS CONTROL ROM FIELD DEFINITIONS

```
ALU_9.SAT[]+PC    "RAMX/Q.AYX/RAMX.SAT.DT,#1,BYX/PC,ALU/A+B"
ALU_9.ADR.9       "RAMX/Q.AYX/RAMX,BYX/RDMX,RAMX,9,ALU/ADR"
ALU_9.SOR.K[]     "RAMX/Q.AYX/RAMX,AMX/#1.BYX/KYA.ALU/XOR"
ALU_9.XOR.LC      "RAMX/Q.AYX/RAMX,BYX/LC.ALU.XOR"
ALU_R.]           "SPO.R/LOAD.LAB,SPO.RAB/#1.AMX/LA,ALU/A"
ALU_R[].AND.K[]   "SPO.R/LOAD.LAB,SPO.RAB/#1,AYX/LA.AYX/R2.BYX/KYA,ALU/AND"
ALU_R[].ANDNOT.MASK   "SPO.R/LOAD.LAB.SPO.RAB/#1.AMX/LA,BYX/MASK.ALU/ANDNOT"
ALU_R[DST]        "SPO.AC/LOAD.LAB.SPO.ACNTL/DST.DST,AYX/LA,ALU/A"
ALU_R[].OR.K[]    "SPO.R/LOAD.LAB.SPO.RAB/#1,AMX/LA.AYX/R2.BYX/KYA,ALU/OR"
ALU_R[].ORNOT.K[] "ALU/ORNOT,AMX/LA.BYX/KYX.SPO.R/LOAD.LAB,SPO.RAB/#1,AMX/LA"
ALU_R[].XOR.K[]   "SPO.R/LOAD.LAB,SPO.RAB/#1.AYX/LA.AYX/R2,BYX/KYA,ALU/XOR"
ALU_RC[]          "SPO.R/LOAD.LC,SPO.RC/#1.BMX/LC.ALU/B"
ALU_RC(SC)        "SPO/LOAD.LC.SC,BMX/LC.ALU/B
ALU_R(SP1)+K[].RLOG   "SPO.AC/LOAD.LAB,SPO.ACN/SP1.SP1.AMX/LA.KYB/#1,BMX/KMX,ALU/A+B.RLOG"

CACHE_D[]         "VAX/NOP.MCT/WRITE.V.ACHK,DT/#1,DK/NOP
CACHE[].D         "VAX/NOP,MCT/WRITE.V.ACHK,MSC/#1,DK/NOP"
CACHE_D.INST.DEF      VAX/NOP.MCT/WRITE.V.ACHK.DT/INST.DEF,DK/NOP"
CACHE_D[].NOCHK   "VAX/NOP,MCT/WRITE.V.NOCHK.DT/#1,DK/NOP"
CACHE.P_D[]       "VAX/NOP,MCT/WRITE.P,DT/#1.DK/NOP"
CACHE_D[].LK      "VAX/NOP,MCT/LOCKWRITE.V.ACHK.DT/#1.DK/NOP"
;D_0... THRU D_CACHE.

D_D               "DK/CLR"
D_D+D             "AMX/RAYX.DXT.DT/LONG.BDYX/D.BYX/RDMX,ALU/A+B.SHF/ALU,DK/SHF"
D_D+K[]+1         "AMX/RAYX.DXT.DT/LONG.KYX/#1,BYX/KYA,ALU/A+B+1.SHF/ALU,DK/SHF"
D_D+LC+1          "AMX/RAYX.DXT.DT/LONG.BYX/LC.ALU/A+B+1,SHF/ALU,DK/SHF"
D_D-Q             "AMX/RAYX.DXT,DT/LONG.RBMX/Q,BYX/RBMX.ALU/A-B.SHF/ALU,DK/SHF"
D_ACCEL.SYNC      "DK/ASSE.ACT/SYNC"
D_ALU             "SHF/ALU.DK/SHF"
D_ALU.LEFT        "SHF/LEFT.DK/SHF"
D_ALU.LEFT2       "SHF/ALU.DT.DT/LONG.DK/SHF"
D_ALU.LEFT3       "SHF/LEFT3,DK/SHF"
D_ALU.RIGHT       "SHF/RIGHT.DK/SHF"
D_ALU.RIGHT2      "SHF/RIGHT2.DK/SHF"
D_ALU(FRAC)       "SHF/ALU.DK/SHF.FL"
D_BLANK           "D_R[.SC]
D[]_CACHE         "VAX/NOP,MCT/READ.V.RCHK.DT/#1.DK/NOP"
D_CACHE[]         "VAX/NOP,MCT/READ.V.RCHK,MCT/#1,DK/NOP"
D_] CACHE.IBCHK   "VAX/NOP,MCT/READ.V.IBCHK.DT/#1.DK/NOP
```

```
D_CACHE.TEST.DFB        'CAB/NCP,WCT/READ,V.TBCHK,DT/.WST,DBF,DR/NCP'
D_CACHE.LRU[]           'WAR/NCP,WCT/CLOCKREAD,V.WCHK,CPC/$1,DR/NCP'
D[]_CACHE.TAG           'WAR/NCP,WCT/CLOCKREAD,V.WCHK,DT/WT,DR/NCP'
D[]_CACHE.NOCHK         'WAR/NCP,WCT/READ,V.NOCHK,DT/$,,DR/NCP'
D[]_CACHE.P             'WAR/NCP,WCT/READ,P,DT/$1,DR/NPF'
D[]_CACHE.NCHK          'WAR/NCP,WCT/READ,V.NCHK,DT/$1,DR/NDF'
D_CACHE.WCHA[]          'WAR/NCP,WCT/READ,V.WCHK,WSC/$1,DR/NCP'
;D_DAL... THRU D_D...

D_DAL.WORD              'DX/DA,,$0'
D_DAL.$0                'DR/DA,,$0'
D_D.DXT[]               'RAMA/D,AMX/RAMX,DXT,DT/$1,ALU/A,SHF/ALU,DR/SHF'
D_D.DXT[].XORNOT.R[]    'RAMA/D,AMX/RAMX,DXT,DT/$1,KMX/$2,EMX/KMX,ALU/ANDNOT,SHF/ALU,DR/SHF'
D_D.DXT[]+R[]           'RAMA/D,AMX/RAMX,DXT,DT/$1,KMX/$2,EMX/KMX,ALU/A+B,SHF/ALU,DR/SHF'
D_D.DXT[].DP.Q          'RAMA/D,AMX/RAMX,DXT,DT/$1,-PMX,Q,EMX/BENA,ALU/OR,SHF/ALU,DR/SHF'
D_D.DXT[]-Q             'F_U/A+B,AMX/RAMX,DXT,DT,T,-PMX,B,XX,PMX/Q,Q,ALU'
D_D.DXT[]+0+1           'RAMA,D,AMX/RAMX,DXT,DT/$1,YCX,PMX,ALU,A+B+1,D_A,U'
D_D.DXT[].XOR.D         'DR/SHF,ALU/XOR,SHF/ALU,AMX/RAMX,DXT,DT/$1,PMX,D,DT/$1,EMX/D,EMX/REMX'
D_D.DXT[].XOR.R[]       'RAMX/D,AMX/RAMX,DXT,DT/$1,SPD,R/LOAD,LD.SPD,RD/$1,EMX/LC,ALU/XOR,SHF/ALU,DR/SHF'
D_D.AND.R[]             'KMX/D,AMX/RAMX,DXT/$1,EMX/KMX,ALU/AND,SHF/ALU,DR/SHF'
D_D.AND.R[]..SFTR       'RAMX/D,AMX/RAMX,KMX'$1,EMX/KMX,ALU/AND,SHF/ALU,DT,DT/LINK,DR/SHF
D_D.AND.R[].DP          'RAMX/D,AMX/RAMX,KMX'$1,EMX/KMX,ALU/AND,SHF/RIGHT,DR/SHF'
D_D.AND.LC              'RAMX/D,AMX/RAMX,EMX/LC,ALU/AND,SHF/ALU,DR/SHF'
D_D.AND.EMX             'RAMX/D,AMX/RAMX,EMX/EMX,ALU/AND,SHF/ALU,DR/SHF'
D_D.AND.Q               'RAMX/D,AMX/RAMX,EMX/Q,EMX,KMX,ALU/AND,SHF/ALU,DR/SHF'
D_D.AND.RC[]            'RAMX/D,AMX/RAMX,SPC,R/LOAD,LD.SPD,RD/$1,EMX/LC,ALU/AND,SHF/ALU,DR/SHF'
D_D.ANDNOT.R[]          'RAMX/D,AMX/RAMX,KMX,R,,EMX,KMX,ALU/ANDNOT,SHF/ALU,DR/SHF'
D_D.ANDNOT.LC           'RAMX/D,AMX/RAMX,KMX,R,,D,ALU/ANDNOT,SHF/ALU,DR/SHF'
D_D.ANDNOT.MSK          'DR/SHF,ALU/ANDNOT,AMX/RAMX,EMX/D,KMX/$FFFF,4X/.4,SHF/ALU'
D_D.ANDNOT.R            'RAMX/D,AMX/RAMX,RDMX/D,EMX/EMX,4,U/ANDNOT,SHF/ALU,DR/SHF'
D_D.ANDNOT.RC[]         'RAMX/D,AMX/RAMX,SPC,R/LOAD,LD.SPD,RD/$1,EMX/LC,ALU/ANDNOT,SHF/ALU,DR/SHF'
D_D[]R4C                'RAMX/D,AMX/RAMX,ALU/A+B+1,SHF/ALU,DR/SHF,FL'
D_D[]R[]                'RAMX/D,AMX/RAMX,4X/$2,EMX,KMX,ALU/$1,SHF/ALU,DR/SHF'
D_D+K[]                 'RAMX/D,AMX/RAMX,4X/.5,,EMX,KMX,ALU/A+B,SHF/ALU,DR/SHF'
D_D+K[]                 'RAMX/D,AMX/RAMX,4X/$1,EMX/KMX,ALU/A+B,SHF/ALU,DR/SHF'
D_D+K[]+1               'RAMX/D,AMX/RAMX,4X/$1,EMX/KMX,ALU/A+B+1,SHF/ALU,DR/SHF'
D_D+LB                  'RAMX/D,AMX/RAMX,EMX/.B,,ALU/A+B,SHF/ALL,DR/SHF'
D_D+LC                  'RAMX/D,AMX/RAMX,EMX/LC,ALU/A+B,SHF/ALL,DR/SHF'
D_D+LC                  'RAMX/D,AMX/RAMX,EMX/LC,ALU/A+B,SHF/ALL,DR/SHF'
D_D+LC!PSL.C            'RAMX/D,AMX/RAMX,EMX/LC,ALU/A+B+PSL.C,SHF/ALU,DR/SHF'
```

```
D_D.LEFT          'DX/LEFT'
D_D.LEFT2         'DX/LEFT2'
D_D]MASK          'RAMX/D,AMX/RAMX,SHF/MASK ALU/#1,SHF/ALU,DX/SHF'
D_D.DR.ASCII      'D D.DR.X[.3D]'
D_D.DR.X[         'RAMX/D,AMX/RAMX,MMX/#1.SMX, AMX.ALU/DR,SHF/ALU,DX/SHF'
D_D.DPHOT,MASK    'RAMX/D,AMX/RAMX,SMX/MASK,ALU/LATCH',SHF/ALU,DX/SHF'
D_D.OR.PSAC       'DX/SHF,ALU/OR,AMX/RAMX.PATH/D.SMX/AMX,MMX/.1,SHF/ALU'
D_D.OR.PSAV       'DX/SHF,ALU/OR,AMX/RAMX.PATH D.SMX/AMX,MMX/.0,SHF/ALU'
D_D.OR.Q          'RAMX/D,AMX/RAMX,AMX/Q SMX/RAMX,ALU/OR,SHF/ALU DX/SHF'
D_D.OR.PC[]       'RAMX/D,AMX/RAMX,SPD.R/.DAC.LC.SPC.RC/P1,SMX/LC.ALU/OR,SHF/ALU,DX/SHF'
D_D]Q             'RAMX/D,AMX/RAMX,-AMX/Q.SMX/RAMX,ALU/#1.SHF/ALU.DX/SHF'
D_D]Q             'RAMX/D,AMX/RAMX,-AMX/Q.SMX/RAMX,ALU/A+B.SHF/ALU.DX/SHF'
D_D-Q             'RAMX/D,AMX/RAMX,-AMX/Q.SMX/RAMX,ALU/A-B.SHF/ALU,DX/SHF'
D_D+Q+1           'RAMX/D AMX/RAMX,AMX/Q.SMX/RAMX,ALU/A+B+1,SHF/ALU,DX/SHF'
D_D-Q-1           'RAMX/D AMX/RAMX,-AMX/Q.SMX/RAMX,ALU/A-B-1,SHF/ALU,DX/SHF'
D_D.RIGHT         'DX/RIGHT'
D_D.RIGHT2        'DX/RIGHT2'
D_D.RIGHT[S]      'RAMX/D.SMX/RAMX,ALL/2,SHF/RIGHT,DX/SHF'
D_D.SWAP          'DX/SWAP.SWAP'
D_D.SXT[]         'RAMX/D.AMX/RAMX.SXT,ST/#1,ALL/A,SHF/ALL,DX/SHF'
D_D.SXT[].RIGHT   'RAMX/D.AMX/RAMX.SXT,ST/#1,ALL/A,SHF/RIGHT,DX/SHF'
D_D.XOR.R[]       'RAMX/D.AMX/RAMX,AMX/R',RAMX/RAM),ALU/XOR,SHF/ALU,DX/SHF'
D_D.XOR.LC        'RAMX/D.AMX/RAMX,BMX/LC,ALU/XOR,SHF/ALU,DX/SHF'
D_D.XOR.A         'RAMX/D.AMX/RAMX,RAMX/D,BMX/RAMX,ALU/XOR,SHF/ALU,DX/SHF'

;D_INT.SLW... THRU D_PC...

D_INT.SLW         'NOT/READ.INT.SLW,DX/NOP'
D_R[              'RMX/#1,DMX/RMX,ALL/B.SHF/ALL,DX/SH'
D_R[].RIGHT       'RMX/#1,DMX/RMX,ALL/B,SHF/RIGHT,DX/SHF'
D_R[].RIGHT2      'RMX/#1,DMX/RMX,ALL/B,SHF/RIGHT2,DX/SHF'
D_L4              'AMX/L4,ALL/A,SHF/ALL,DX/SHF'
D_L4,AND.R[]      'AMX/L4,RMX/#1,BMX/RMX,ALL/A&B,SHF/ALL,DX/SHF'
D_L4#DIPSL.C      'AMX/L4,BMX/D,BMX/-BMX,ALL/A-B+PSL.C,SHF/ALL,DX/SHF'
D_L4-D            'DX/SHF,ALL/A-B.AMX/L4,BMX/RAMX,RBMX/D,SHF/ALU'
D_L4[FRAC]        'AMX/L4,ALL/A,SHF/ALL,DX/SHF.FL'
D_L4-R[]          'AMX/L4,RMX/#1,-MX/4M4,ALL/A-B,SHF/ALL,DX/SHF'
D_L4-RIGHT        'AMX/L4,ALL/A,SHF/RIGHT,DX/SHF'
D_L8              'BMX/L8,ALL/B,SHF/ALL,DX/SHF'
D_L8.PC           'BMX/PC,DX.L8,ALL/B,SHF/ALL,DX/SHF'
```

```
D_LC            'RNA/LC,A1,A0,BUS/ALU,DA/SHF'
D_LC(FRAG)      'EMA/LC,A1,A0,BUS/ALU,DA/SHF'
D_MQ.E          'RAM/Q,RNA/Q,A1,A0,NOT.Q/Q,ALU,DA/SHF'
D_MQ.MQ[]       'RAM/Q,RNA/Q,A1,A0,NOT.Q/Q,ALU/ONES,ALU,DA/SHF'
D_MQ.MQSH       'SHIFT/SHF,RAM/Q,RNA/Q,A1,A0,NOT.Q/Q,ALU/ONES,SHF/ALU,DA/SHF'
D_MQ1.Q         'RAM/Q,RNA/Q,A1,A0,NOT.Q/Q,ALU,DA/SHF'
D_MQ1.Q[]       'LM.SHIFT,RAM/Q,A1,A0,NOT.Q/Q,ALU,DA/SHF'
D_PACK.FP       'SHIFT/SHF,RAM/Q,A1,A0,NOT.Q/Q,ALU,DA/SHF'
D_PACK.FP.LEFT  'SHIFT/SHF,RAM/Q,A1,A0,NOT.Q/Q,ALU,DA/SHF'
D_SC            'SHIFT/SHF,A1,A0/B.SHIFT,ALU,DA/SHF'
D_SC.FIT        'SHIFT/SHF,A1,A0/B.SHIFT/LEFT,DA/SHF'
IE_D.-          'SHIFT/SHF,A1,A0/B.SHIFT/LEFT,DA/SHF'

D_0             'DA,0'
D_0.DAT.[]      'RAM/Q,RNA/Q,A1,A0,NOT.Q/Q,ALU,DA/SHF'
D_0.MA.R[]      'RAM/Q,RNA/Q,A1,A0,RAM,ALU/RAM,SHF/ALU,DA/SHF'
D_0.MA.SC       'RAM/Q,RNA/Q,A1,A0,SHF/ALU,DA/SHF'
D_0.ANT.MASK    'RAM/Q,RNA/Q,A1,A0,NOT.Q/Q,SHF/ALU,DA/SHF'
D_0.ANXNOT.D    'RAM/Q,RNA/Q,A1,A0,RAM,ALU/ANDNOT,SHF/ALU,DA/SHF'
D_0.ANXNOT.MASK 'RAM/Q,RNA/Q,A1,A0,RAM,ALU/ANDNOT,SHF/ALU,DA/SHF'
D_0.ANXOT.FRAC  'RAM/Q,RNA/Q,A1,A0,RAM,ALU/ANDNOT,SHF/ALU,DA/SHF'
D_0.ANXOT.MASK  'RAM/Q,RNA/Q,A1,A0,RAM,ALU/ANDNOT,SHF/ALU,DA/SHF'
D_0.ANXOT.MASK2 'RAM/Q,RNA/Q,A1,A0,RAM,ALU/ANDNOT,SHF/ALU,DA/SHF'
D_0.ANXE.R[]    'RAM/Q,RNA/Q,A1,A0,RAM,ALU/ANDNOT,SHF/ALU,DA/SHF'
END.D=0         'RAM/Q,RNA/Q,A1,A0,NOT.Q/Q,SHF/ALU,DA/SHF'
D_0.MD          'RAM/Q,RNA/Q,A1,A0,NOT.Q/Q,SHF/ALU,DA/SHF'
D_0.-           'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B,SHF/ALU,DA/SHF'
D_0.-.I         'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.1D          'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.1D.D        'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.1D.D.NOCI   'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.-.[]        'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.-.[]-1      'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.LB          'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.LB.D        'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.LEFT        'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.OR.-[]      'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.ORNOT.MASK  'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
D_0.PSW         'RAM/Q,RNA/Q,A1,A0,RAM,ALU/A=B-1,SHF/ALU,DA/SHF'
```

```
D_[].DA.SC[]      'RAYA/S,ANX/RAYA,S_L,>,GLL,LC,SPD.RC/#1,BMX/LC,ALU/OR,SHF/ALU,DK/SHF'
D_[+RC            'RAYA/S,ANX/RAYA,BMX/RC,ALU/A+B,SHF/ALU,DK/SHF'
D_[-RCSU          'RAYA/S,ANX/RAYA,BSD/C,YOC/R,AS,RLCG,ALU/A-B,SHF/ALU,DK/SHF'
D_[.RIGHT         'RAYA/S,ANX/RAYA,ALU/A,SHF/RIGHT,DK/SHF'
D_[.RIGHT2        'RAYA/S,ANX/RAYA,ALU/A,SHF/RIGHT2,DK/SHF'
D_[.SXT[]         'RA/A/Q,AMX/RAMX,SXT,BT/#1,ALU/A,SHF/ALU,DK/SHF'
D_[.XOR.SC[]      'RANX/Q,AMX/RAMX,SPC.R/LOAD.LC.SPD.RC/#1,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF'
,D,A... THRU DAYA_...

D_R[]             'SPD.R/LOAD.LAB,SPD.RAB/#1,ANX/LA,ALU/A,SHF/ALU,DK/SHF'
D_R[].AND.R[]     'SPD.R/LOAD.LAB,SPD.RAB/#1,ANX/LA,BMX/#2,BMX/BMX,ALU/AND,SHF/ALU,DK/SHF'
D_R[].ORNOT.R[]   'LAB_R[#1],ANX/LA,BYA/BMX,BMX/#2,ALU/ORNOT,U_ALL'
D_RD[]            'SPD.R/LOAD.LC.SPD.RC/#1,BYA/LC,ALU/B,SHF/ALL,DK/SHF'
DERD[]_PC         'BYA/PC,ALU/B,SHF/ALU,DK/SHF,SPD.R/ABLE.RC,SPD.RC/#1'
D_RD[SC]          'SPD/LOAD.LC.SC,BMA/LC,ALU/B,SHF/ALU,DK/SHF'
D_R[]|FRAC)       'SPD.R/LOAD.LAB,S_U,"AB/#1,AMX/LA,ALU/A,SHF/ALU,DA/SHF.FL'
D_RLOD            'BYA/D,MOD/READ.RLOD,ALU/B,SHF/ALU,DK/SHF'
D_RLOD.RIGHT      'BYA/D,MOD/READ.RLOD,ALU/B,SHF/RIGHT,DK/SHF'
D_R(PRN-1)        'SPD.AC/LOAD.LAB,SPD.ADR/PRN-1,ANX/LA,ALU/A,SHF/ALU,DK/SHF'
D_R[SC)           'SPD.AC/LOAD.LAB,SPD.ADR/SC,ANX/LA,ALU/A,SHF/ALU,DK/SHF'
D_R[SP1-1)        'SPD.AC/LOAD.LAB,SPD.ADR/SP1-1,ANX/LA,ALU/A,SHF/ALU,DK/SHF'
DSVA_ALU          'VAR/LOAD,SHF/ALU,DA/SHF'
'DSVA_D-4[]       'RAMA/C,AMX/RAMA,BMX/#1,BND/BNX,ALL/A-B,VAA/LOAD,SHF/ALL,DA/SHF'
DSVA_D+LC         'RAYA/C,AMX/RAM+,BMX/LC,ALU/A+B,VAR/LOAD,SHF/ALU,DK/SHF'
DSVA_D+Q          'D_C-Q,VAA/LOAD'
DSVA_LA           'AYX/LA,ALU/A,VAR/LOAD,SHF/ALU,DA/SHF'
DAVA_L6           'BYA/LB,ALU/B,VAR/LOAD,SHF/ALU,DA/SHF'
DAVA_Q            'RAYA/Q,AMX/RAMA,ALU/A,VAA/LOAD,DK/Q'
DSVA_D+LB,PC      'RAYA/Q,AMX/RAYA,BMX/PC,C+1,D,ALU/A+B,VAR/LOAD,SHF/ALU,DA/SHF'
'EA:[]_. . THRU FE_...

EALU_DIEXP)       'RA/A/Q,AMX/RAYA,EBMX/RAYA.EXP.EALU/B'
EALU_FE           'EBYX/FE,YA,U/B'
EALU_4:]          'BNX.#1,EBYX/AYA,EALU/B'
EALU_R[]EXP)      'SPD.R/LOAD.LAB.SPD.RAB/R',AMX/LA,EBMX/AMX.EXP.EALU/B'
EALU_SC           'EALL/A
EALU_SC.AMENOT.A[]        'MA/#1,EBMA/BNB.EALL/AANOT'
EALU_SC+FE        'EBYA/FE.LA.U/A-B
EALU_SC-FE        'EBYA/FE,YA,U/A-B
EALU_SC+A']       'BNX/#1,EBYX/AYA,EALU/A+B'
```

```
EALU_SC-N[]     "ALU/B,EBNX/ENX,SA-U/A-B
EALU_STATE      "EALU/A,MSC/LOAD,STATE"

FE_D(A)         "ANX/BANC,OAT,U/LDA,EBNX/ATA,ESC,EALU/B,FEA/LOAD"
FE_D(EXP)       "EANX/D,ATA/BANX,EBYA/ANX,EXP,EALU/B,FEK/LOAD"
FE_FA_U         "FEA/_D4D"
FE_K[]          "ANX/A1,EENX/ENX,FA_U/B,FEK/LOAD"
FE_LA(EXP)      "ANX/_A,EBNX/ANX,EXP,EALU/B,FEA/_LOAD"
FE_MASSIGO-LA(EXP)  "ANX/_A,EBYA/ATA,EXP,EALU/MASS,A-B,FEK/LOAD"
FE_D(EXP)       "EANX/Q,ANA/BANX,EBNA/ANX,EXP,EALU/B,FEK/LOAD"
FE_K[](EXP)     "S-D,R/LOAD,LAB,SPC,RAB/A,ATA/LA,EBNX/ANX,EXP,EALU/B,FEA/LOAD"
FE_SC           "EALU/A,FEA/_D4D"
FE_SC.ANDNOT.FE         "ENX/FE,FA_U/ANDNOT,FEK/LOAD"
FE_SC.ANDNOT.K[]        "ENX/A1,EBYX/ENX,EALU/ANDNOT,FEK/LOAD"
FE_SC-1         "EALU/A-1,FEK/LOAD"
FE_SC+FE        "EBNX/FE,EALU/A+B,FEK/LOAD"
FE_SC-FE        "EBNX/FE,EALU/A-B,FEK/LOAD"
FE_SC_K[]       "ANX/A1,EENX/ENX,FA_U/B,FEK/_LOAD,SHA/EALU,SCK/LOAD"
FE_SC+K[]       "ANX/A1,EENX/ENX,FA_U/A+B,FEK,LOAD"
FE_SC+K[]       "ANX/A1,EENX/ENX,FA_U/A+B,FEK,LOAD"
FE_SC+LA(EXP)   "ANX/_A,EBNX/ANX,EXP,EALU/A+B,FEK/LOAD"
FE_SC-LA(EXP)   "ANX/_A,EBNX/ANX,EXP,EALU/A-B,FEK/LOAD"
FE_SC.OR.K[]    "EALU/OR,EBYA/ATA,ENX/A1,FEA/LOAD"
FE_SC-SHFT.VAL  "EBNX/SHF,VAL,EALU/A-B,FEK/LOAD"
FE_SHFT.VAL     "EBNX/SHF,VAL,EALU/B,FEK/LOAD"

;(D_,,, YMSU LL_,,,

FD[]_D          "SLD/A1,TE.ANX,LD.ADDR/A1"
LD_D2N3,SYNC    "SLD/A1,TE.ANX,ADS/LDA,RNX/SP1.CON"
LD_D,SYNC       "SLD/A1,TE.ANX,ADS/LDA,RNX/SP1.CON,NEX/SYNC"
LD(SC)_D        "SLD/A1,TE.SC"

K[]             "KMX/A1"

LA_RA[]         "SPD,AS/_D4C,LA,SPC,RAB/A1"
LA_R(DSY)PUB_R(SRC)     "SPC,AS/LOAD,LAB,SPD,ACN'1/DST_SRC"
LA_R(SP1)(D_R(SP1))     "SPD,AS/LOAD,LAB,SPD,ACN/SP0,SP1"
LAB_A1&RS_|_D           "ALU_D1A1,,LAB_A1&-2,A1,,ALU"
LAB_A1&A2[]_ALU  SPC,R/LOAD,LAB\,ASTTC,BL,SPD,BC/A1,SHF/ALU"
```

```
LAB_R1&R2[]_A.U.RIGHT2   "SPD.R,LOAD.LAB1.WRITE.RC,SPD.RC/#1,SHF/RIGHT*2"
LAB_R1&RC[]_D.DST[]&R[]  "ALU_D.DST[#2]-R[#3],LAB_R1&RC[#1]_ALU"
LAB_R1&RC[]_D+UC         "ALU_D+UC,LAB_R1&RC[#1]_ALU"
LAB_R1&RC[]_D-R[]        "ALU_D-R[#2],LAB_R1&RC[#1]_ALU"
LAB_R1&RC[]_D+D  "SPD.R/LOAD.LAB1.WRITE.RC,SPD.RC/#1,ALU/A+B,&MA/R&MX.DST.DT/LONG.BMX,SHF/ALU"
LAB_R1&RC[]_D+.D+1       "ALU/A+B+1,&MX/R&MX.DST.DT/LONG.BMX/LC.SPD.R/LOAD.LAB1.WRITE.RC,SPD.RC/#1,SHF/ALU"
LAB_R[]          "SPD.R/LOAD.LAB.SPC.RAS/#"
LAB_R(DST)       "S.D.AC/.LOAD.LAB.SPD.AC#1/DST.DST"
LAB_R(SC)        "S.D.AC/.LOAD.LAB.SPD.AC#/SC"
LAB_R(SP1)       "SPD.AC/.LOAD.LAB.SPD.AC#/S-1.SP1"
LC_RC[]          "SPD.R/LOAD.LC,SPD.RC/#1
LC_RC[]&R1_D              .ALU_D, C_RC[#]&R1_ALU"
LC_RC[]&R1_A-R[]          .ALU_A-R[#3],LC_RC[#1]&R1_ALU"
LC_RC[]&R1       "SPD/.LOAD.LC.SC
LC_RC[]&R1_A_1   "SPD.R/LOAD.LC.WRITE.RAB1,SPD.RC/#1,SHF/ALU"
LC_RC[]&R1_(A+.S1).LEFT  "&MX/.A,SNX.LB,ALU/A+B.SHF/LEFT,SPD.R/.LOAD.LC.WRITE.RAB1,SPD.RC/#1"
LC_RC[]&R1_(A-.S1).LEFT  "&MX/.A,SNF/LB,ALU/A-B.SHF/LEFT,SPD.R/.LOAD.LC.WRITE.RAB1,SPD.RC/#1"
LC_RC[]&R1_(A-4.)        "SPD.R/LOAD.LC.WRITE.RAB1,SPD.RC/#1,A-B/ALU,#1U/A+B.&MX.LA.BY4/&MX.&MX/&2"
LC_RC[]&R1_LB    "ALU_LB.LC_RC[#1]&R1_ALU"
LC_RC[]&R1_D     "SPD.R/LOAD.LC.WRITE.RAB1,SPD.RC/#1,SHF/ALU.ALU/A,&NX/R&MX,R&MX/2"
;PC_... THRU PC&VA_...

PC_PC+1          "PCR/PC+1"
PC_PC+2          "PCR/PC+2"
PC_PC+4          "PCR/PC+4"
PC_PC+N          "PCR/PC+N"
PC_C+PC          "ALU/A+C,CAR/LOAD,PCR/PC,VA.BYX/PC.&Y4/RMNX,R&MX/2
PC_VA            "PCR/PC_VA"
PC&VA_ALU        "CAR/LOAD,PCX/PC_VA"
PC&VA_D          "RAMA/D,&NX/R&MX,ALU/A,VAX/.LOAD.PCR/PC_VA"
PC&VA_D.DST[]    "R&MX/D,&NX/R&MX.DST.DT,#1,ALU/A,VAX/.LOAD.PCR/PC_VA"
PC&VA_D.DST[]+PC         "RAMA/D,&MX/R&MX.DST.DT,#1,BYX/PC.ALU/A+1,VAX/.LOAD,PCX/PC_VA"
PC&VA_D.DST[]+PC  "&MX/D,&YX/&MX.DST.DT,#1,BYX/PC.ALU/A+B,VAX/.LOAD.PCR/PC_VA"
PC&VA_D+1        "R&YX/D,&NX/&YX+,FM0,#1,ENX/.VX.ALU/A+1,VAX/.LOAD.PCR/PC_VA"
PC&VA_D+A        "R&YX/D,&NX/&YX+,FM0,#1,ENX/.VX.ALU/A+B,VAX/.LOAD.PCR/PC_VA"
PC&VA_D+PC       "R&YX/D,&NX/&YX+,FM0,#1,ENX/PC.ALU/A+B,VAX/.LOAD,PCR/PC_VA"
PC&VA_D[]        "ENX/#1,ENX/+NX.A.1/B.CAR.LOAD,PCX/PC_VA"
PC&VA_PC         "ENX/PC,ALU/B,VAX/.LOAD.PCR/PC_VA"
PC&VA_Q          "R&YX/D,&MX/&YX+,FM0,#1,ENX/.VX.ALU/A+B,VAX/.LOAD.PCR/PC_VA"
PC&VA_Q+D        "R&YX/D,&MX/&YX+,FM0,#1,ENX/.VX.ALU/A+B,VAX/.LOAD,PCX/PC_VA"
```

MICRODIAGNOSTICS CONTROL ROM FIELD DEFINITIONS

```
PC&VA_Q+R[]        "RAMX/Q,AYX/RAMX,BMX/#1,BYX/AYX,ALU/A+B,VAX/LOAD,PCA/PC_VA"
PC&VA_Q+PC         "RAMX/Q,AYX/RAMX,BYX/PC,ALU/A+B,VAX/LOAD,PCA/PC_VA"
PC&VA_Q.SX[]+PC          "RAMX/Q,AMX/RAMX,SXT,DT/BYT,BMX/PC,ALU/A+B,VAX/LOAD,PCA/PC_VA
PC&VA_R[].ANDNOT.K[]  SPO.R/[DOC.LAB,SPO.RAB/K],AMX/...,BMX/#2,BMX/RMX,ALU/ANDNOT,VAX/LOAD,PCA/PC_VA"
PC&VA_RC[]         "SPO.S/LOAD,LC,SPO.RC/#1,BMX/LC,ALU/B,VAX/LOAD,PCA/PC_VA"
PC_VA&A            "PCA/PC_LSA"

;Q_O...  TRY] Q_D ...

Q_0                "QK...R"
Q_C+D              "AMX/RAMX,SXT,DT/LONG,RBYX/D,BMX/RBYX,ALU/A+B,SHF/ALU,QK/SHF"
Q_C+R[]            "AMX/RAMX,SXT,DT/LONG,BMX/#1,BMX/RYX,ALU/A+B,SHF/ALU,QK/SHF"
Q_C+LC             "AMX/RAMX,SXT,DT/LONG,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF"
Q_C+VA.MASK        "AMX/RAMX,SXT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,QK/SHF"
Q_C+PC.RIGHT       "AMX/RAMX,SXT,DT/LONG,BMX/PC,ALU/A+B,RLOG,SHF/ALU,QK/SHF"
Q_D+Q              "AMX/RAMX,SXT,DT/LONG,RBYX/Q,BYX/RBMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_ACCELSSYNC       "QK/ACCEL,ACC/SYNC"
Q_ALU              "SHF/ALU,QK/SHF"
Q_ALU.LEFT         "SHF/LEFT,QK/SHF"
Q_ALU.OT.TO        "SHF/ALU,OT,DT/LONG,QK/SHF"
Q_ALU.RIGHT        "SHF/RIGHT,QK/SHF"
Q_ALU.RIGHT2       "SHF/RIGHT2,QK/SHF"
Q_ALU(FPAC)        "SHF/ALU,QK/SHF,FL"
Q_D                "QK/D"
Q_D(FRACT1B)       "REMX/D,BYX/RBMX,ALU/B,SHF/ALU,QK/SHF,FL"
Q_D.OR.R[]         "RAMX/D,AYX/RAMX,DXT,DT/#1,ALU/A,SHF/ALU,QK/SHF"
Q_D.OR.R[].R[].LEFT    "RAMX/D,AYX/RAMX,DXT,DT/#1,BYX/#2,BYX/RBMX,ALU/A+B,SHF/LEFT,QK/SHF"
Q_D.AND.K[]        "RAMX/D,AYX/RAMX,KXT,BYX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_D.AND.K[].RIGHT          "RAYX/D,AMX/RAMX,BMX/#1,BMX/AYX,ALU/AND,SHF/RIGHT,QK/SHF"
Q_D.AND.K[].RIGHT2         "RAYX/D,AYX/RAMX,BMX/#1,BMX/AYX,ALU/AND,SHF/RIGHT2,QK/SHF"
Q_D.ANDNOT.RC[]    "RAMX/D,AMX/RAYX,SPO.R/LOAD,LC,SPO.RC/#1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF
Q_D.AND.RC[]       "RAYX/D,AMX/RAYX,SPO.R/LOAD,LC,SPO.RC/#1,BMX/LC,ALU/AND,SHF/ALU,QK/SHF"
Q_DEC.CON          "QK/DEC.CON"
Q_D+R[]            "RAYX/D,AMX/RAYX,BMX/#1,BMX/RYX,ALU/A+B,SHF/ALU,QK/SHF"
Q_D+R[].LEFT       "RAYX/D,AMX/RAYX,BMX/#1,BMX/RYX,ALU/A+B,SHF/LEFT,QK/SHF"
Q_D-R[]            "RAYX/D,AMX/RAYX,BMX/#1,BMX/RYX,ALU/A-B,SHF/ALU,QK/SHF"
Q_D+LC             "RAYX/D,AMX/RAYX,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF"
Q_D-LC             "RAYX/D,AYX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_D.LEFT3          "RAYX/D,AYX/RAMX,ALU/A,SHF/LEFT3,QK/SHF"
Q_D.OR.R[]         "RAYX/D,AYX/RAMX,BMX/#1,BYX/AYX,ALU/OR,SHF/ALU,QK/SHF"
```

```
Q_D.JR.RC[]       "RAMA/C,A,A/RAMA,SP_,B/LOAD,LE,GND,RC/#1,BYA/LC ALU/ZR,SHF/ALU,QA/SHF"
Q_D=0             "RAMA/C,A,A/RAMA,,,B/A,BYA/ZERO,ALU/A=B,SHF/ALU,QA/SHF"
Q_D.RIGHT         "RAMA/C,A,A/RAMA,ALU/A,SHF/RIGHT,QA/SHF"
Q_D.RIGHT2        "RAMA/C,A,A/RAMA,ALU/A,SHF/RIGHT2,QA/SHF"
Q_D.SXT[]         "RAMA/C,A,A/RAMA,,,B/#1,ALU/A,SHF/ALU,QX/SHF"
Q_D.XOR_Q         "QA/SHF,ALU/XOR,AMA/RAMA,RAMA/C,A,A/RAMA/RAMA,RBMA/D,SHF/ALU"
;Q_D... THRU Q_PC...

Q_IB.BC=SI        "IBC/BCSET,QA/FD,MST/ALL,QA,IB,RE=4E"
Q_IB.DATA         "QA/FD,MST/ALLCA,IB,RE=4E"
Q_IC[]            "CIC,READ,AMA,ID,AOUR/#1,QA,IC"
Q_ICISC)          "CIC/READ,BC,QR/IC"
Q_K[]             "KNX,#1,ANX,KNX,A,A,B,SHF/A,ALU,QK/SHF"
Q_K[].DIA         "KNX,#1,DIX=KNX,A,A/B,SHF/A,ALU,DT,DT/DISP,DSP,QA/SHF"
Q_K[].RIGHT       "KNX,#1,DIX/KNX,A,A/B,SHF/RIGHT,QK/SHF"
Q_K[].RIGHT9      "KNX,#1,BIX/BNX,ALU/B,SHF/RIGHT9,QK/SHF"
Q_L4              "ANX/L4,ALU/A,SHF/ALU,QA/SHF"
Q_L4.AND_K_]      "ANX/L4,KMA/#1,BIX/BYA,ALU/AND,BUF/ALU,QA/SHF"
Q_L4.ANDNOT.RC[]          "AMX/L4,SPL,B,ISAM,IC,AMA,RC/#1,BYA/LC,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_L4=K[]          "AMX/L4,BMA/B,IMX,,IX,A,A,/A=B,SHF/ALU,QA/SHF"
Q_L4=R[]          "AMX/L4,BMA/B,IMX,,IX,A,A,/A=B,SHF/ALU,QA/SHF"
Q_L4+Q            "AMX/L4,RIMX/Q,IMX,,BYA,ALU/A=B,SHF/ALL,QK/SHF"
Q_LB              "BMA/LB,ALU/B,SHF/ALU,QA/SHF"
Q_LC              "BMA/LC,ALU/B,SHF/ALU,QA/SHF"
Q_NOT.Q           "AMX/Q,AMA/RAMX,A,A/NOTA,SHF/ALU,QK/SHF"
Q_NOT.R[]         "LA RA[#1,QA/,A,A,/NOTA,Q_ALU"
Q_PAGR.HF         "BMA/ARCHID,IC,ALU/R,SHF/ALU,QA/SHF"
Q_PC              "BMA/PC,ALU/B,SHF/ALU,QK/SHF"
;Q_P... THRU QSVA_...

Q_Q.BX[]=4[]      "RAMA/C,A,A/RAMA,CAI,DT/#1,KNX/#2,GNX/BMX,ALL/A=B,SHF/ALU,QA/SHF"
Q_Q.BX[].LEFT     "RAMA/C,A,A/RAMA,CAI,DT/#1,ALU/A,SHF/LEFT,QK/SHF"
Q_Q.BXT[],Q4,Q    "+AMA/C,A,A/RAMA,CAI,DT/#1,BBYA,B,BMX/ABYA,ALU/OR,SHF/ALU,QK/SHF"
Q_Q.AND.K[]       "RAMA/C,A,A/RAMA,BMA/#1,BMA,A,A,ALL/AND,SHF/ALU,QA/SHF"
Q_Q.AND.K[].RIGHT         "+AMA/C,A,A/RAMA,A,A/#1,BYA/AYA,ALU/AND,SHF/RIGHT,QA/SHF"
Q_Q.ANDNOT.Q      "RAMA/C,ANX/AMA,BMA,A,,BMX/ABYA,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_Q.ANDNOT.R[]    "RAMA/C,ANX/RAM,BMA/#1,BMX/ABYA,ALL/ANDNOT,SH=/ALU,QA/SHF"
Q_Q.AND.RC[]      "RAMA/C,ANX/RAM,BMA/#1,BMX/ABYA,ALL/AND,SHF/ALU,QA/SHF"
Q_Q=0             "RAMA/C,ANX/RAMA,BMA/D,SHF/RBYA,ALU/A=B,SHF/ALU,QA/SHF"
Q_QHD             "RA/C/Q,ANX/RAMA,BMA/D,BMX/ABYA,ALU/A=B,SHF/ALU,QA/SHF"
```

MICRODIAGNOSTICS CONTROL ROM FIELD DEFINITIONS

```
Q_Q+B+1          "RAMX/Q,AYX/RAMX,BYX/B,BYX/BMX,ALU/A+B+1,SHF/ALU,OK/SHF"
Q_Q(FRAC)        "RAMX/Q,AYX/RAMX,ALU/A,SHF/ALU,OK/SHF,FL"
Q_Q(FRAC)(3)     "RAMX/Q,BYX/RAMX,ALU/B,SHF/ALU,OK/SHF,FL"
Q_Q+K[]          "RAMX/Q,AYX/RAMX,KYX/K[],BYX/KMX,ALU/A+B,SHF/ALU,OK/SHF"
Q_Q-K[]          "RAMX/Q,AMX/RAMX,KYX/K[],BYX/KMX,ALU/A-B,SHF/ALU,OK/SHF"
Q_Q-K[]-1        "RAMX/Q,AMX/RAMX,KYX/K[],BYX/KMX,ALU/A-B-1,SHF/ALU,OK/SHF"
Q_Q+LC           "RAMX/Q,AMX/RAMX,BYX/LC,ALU/A+B,SHF/ALU,OK/SHF"
Q_Q-LC           "RAMX/Q,AMX/RAMX,BYX/LC,ALU/A-B,SHF/ALU,OK/SHF"
Q_Q-LC-1         "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B-1,SHF/ALU,OK/SHF"
Q_Q.LEFT         "OK/LEFT"
Q_Q-MASK-1       "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU,OK/SHF"
Q_Q.OR.K[]       "RAMX/Q,AMX/RAMX,KMX/#1,BMX/KMX,ALU/OR,SHF/ALU,OK/SHF"
Q_Q.ORNOT.MASK   "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,OK/SHF"
Q_Q+PC           "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,OK/SHF"
Q_Q.RIGHT        "OK/RIGHT"
Q_Q.RIGHT2       "OK/RIGHT2"
Q_Q.SXT[]        "RAMX/Q,AYX/RAMX,KMX/SXT[]/#1,ALU/A,SHF/ALU,OK/SHF"
Q_R[]            "SPO.R/LOAD,LAB,SPO.RAB/#1,AYX/LA,ALU/A,SHF/ALU,OK/SHF"
Q_R[].AND.K[]    "SPO.R/LOAD,LAB,SPO.RAB/#1,AYX/LA,KYX/K[],BYX/KMX,ALU/AND,SHF/ALU,OK/SHF"
Q_R[].AND.K[].RIGHT   "SPO.R/LOAD,LAB,SPO.RAB/#1,AMX/LA,ALU/AND,BMX/KMX,KMX/#2,SHF/RIGHT,OK/SHF"
Q_R[].ANDNOT.K[]      "SPO.R/LOAD,LAB,SPO.RAB/#1,AMX/LA,KMX/#2,BMX/KMX,ALU/ANDNOT,SHF/ALU,OK/SHF"
Q_RC[]           "SPO.R/LOAD,LC,SPO.RC/#1,BMX/LC,ALU/B,SHF/ALU,OK/SHF"
Q_RC[](FRAC)     "SPO.R/LOAD,LC,SPO.RC/#1,BMX/LC,ALU/B,SHF/ALU,OK/SHF,FL"
Q_R[](FRAC)      "SPO.R/LOAD,LAB,SPO.RAB/#1,AYX/LA,ALU/A,SHF/ALU,OK/SHF,FL"
Q_R(PRN).ANDNOT.Q     "SPO.AC/LOAD,LAB,SPO.ACN/PRN,AYX/LA,BMX/Q,BMX/RBYX,ALU/ANDNOT,SHF/ALU,OK/SHF"
Q_R(PRN+1)       "SPO.AC/LOAD,LAB,SPO.ACN/PRN+1,AMX/LA,ALU/A,SHF/ALU,OK/SHF"
Q_R(PRN+1).AND.Q      "SPO.AC/LOAD,LAB,SPO.ACN/PRN+1,AYX/LA,RBYX/Q,BYX/RBMX,ALU/AND,SHF/ALU,OK/SHF"
Q_R(PRN+1).AND.K[]    "SPO.AC/LOAD,LAB,SPO.ACN/PRN+1,AMX/LA,KMX/#1,BMX/KMX,ALU/AND,SHF/ALU,OK/SHF"
Q_SC             "ALU/B,BYX/KMX,KMX/SC,SHF/ALU,OK/SHF"
QAVA_ALU         "VAK/LOAD,SHF/ALU,OK/SHF"
QAVA_Q           "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,SHF/ALU,OK/SHF"
QAVA_Q+LC        "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,OK/SHF"
QAVA_LA          "AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,OK/SHF"
QAVA_Q+LR.PC     "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD,SHF/ALU,OK/SHF"
[illegible]      "PACK"

R[]_Q            "SPO.R/WRITE,RAB,SPO.RAB/#1,AYX/RAMX,CAT,DT/LONG,ALU/A,SHF/ALU"
R[]_Q+B          "AYX/RAMX,CAT,DT/LONG,RAMX/Q,BYX/RBYX,ALU/A+B,SHF/ALU,SPO.R/WRITE,RAB,SPO.RAB/#1"
R[]_Q+K[]        "AYX/RAMX,CAT,DT/LONG,KMX/K[],BYX/KMX,ALU/A+B,SHF/ALU,SPO.R/WRITE,RAB,SPO.RAB/#1"
R[]_Q+LC         "AYX/RAMX,CAT,DT/LONG,BMX/LC,ALU/A+B,SHF/ALU,SPO.R/WRITE,RAB,SPO.RAB/#1"
```

```
R[]_D+L01'        "R/A/BANX.CAT,01/LONG,END,,,,/A-B+',SHF/ALU.SPD,R/WRITE.RAB.SPD,,/A$/01"
R[]_D+Q           "R/A/BANX.CAT,07/LONG,RSD//Q.E/A/BDX ALU/A+B.SHF/ALU,SPC.B/RAIE.RAB,SPD.RAB/01"
R[]_D+1           "R/A/BANX.CAT,07/LONG.END.END,,/1,ALU/A-B.SHF/ALU,SPD.R/WRITE.RAB,SPD.RAB/01"
R[]_ALU           'E+/ALU,D.U.R/ABI E.RAB.SPD.RA$/01'
R[]_ALU.LEFT      'SPL.B/WRITE.RAB,SPD.RAB/01.SH=.LEFT'
R[]_ALU.RIGHT     'SHT/RIGHT,SPD.R/WRITE.RAB,SPD.RAB/01'
R[]_D             'SPC.R/WRITE.RAB,SPD.RAB/01.RAB/D.A/B/BANX.ALU/A.SHF/ALU'
R[]_D.AND.A[]     "SPD.R/WRITE.RAB SPC.RAB/01.ALU/AND.AN//RAJA.BANX/D.BNX/.A/A/B3,SHF/ALU"
R[]_D+1           "SPD.R/WRITE.RAB.SPC.RAB/01.BANX/D.ANX/B+A/A/B3.B7A/BPX,ALU/A+B.SHF/ALU"
R[]_D+M1          "SPD.R/WRITE.RAB.SPC.RAB/01.BANX/D.ANX/B+A/A/B3.B7A/BNX,ALU/A-B.SHF/ALU"
PR.D+K[].RLOG     "SPD.R/WRITE.RAB,SPC.RAB/0B.BANX/D,ANX/B+A/A.A/B3.B7A/SHT,ALU/A+B.RLOG.SHF/ALU"
R[]_D+L0+'        "ALU,D+L0+'.R[#1] A.U
R[]_D.U1.C        'SPD.R/WRITE.RAB.SPD.RA$/0 .A.U/OP,ANX/B+A/A BANX/D.B7A/LL.SHF/ALU'
R[]_D.U1.AND.PP       "SPD.R/WRITE.RAB,SPD.RA$/01,ALU/OP,ANX/B+A/A.BANX/J,BNX/.A/WRITE.LL.SHF/ALU"
R[]_D.D1.Q        'SPE.R/WRITE.RAB,SPC.RA$/0 BANX/E.A7A/B+A/A.+BYA/Q,B7A/B.A/A.ALL/SD.SHF/ALU'
R[]_D+Q           'SPD.R/WRITE.RAB,SPC.RA$/01.BANX/E.A7A/B+A/A.+BYA/Q,B7A/B.A/A.ALL/A+Q.SHF/ALU'
R[]_D+1           'SPE.R/WRITE.RAB,SPC.RA$/01.BANX/E.A7A/B+A/A.+BYA/Q,B7A/B.A/A.ALL/A+Q.SHF/ALU'
R[]_D+Q-1         'SPE.R/WRITE.RAB,SPC.RA$/01.BANX/E.A7A/B+A/A.+BYA/Q,B7A/B.A/A.ALL/A+Q+1.SHF/ALU'
R[]_D[]           'SYA/A7A.A7A/B?.A.U,B.SHF/A.U.SD.A/WRITE.RAB,SPD.RAB/01'
R[]_LA            'SPC R/WRITE.RAB.SPD.RAB 01.A7A/LA.ALL/A,B+/A/1'
R[]_LA.AND.K_     'A7A/LA.BYA/A7A.A7A/BA.ALU/AND.AN/B+A/A.BANX/B.BNX/.A/WRITE.RAB.SPD.R+B/01'
R[]_LA+D          'A7A/LA.+BYA/J.B7A/.BNX,ALU/A+B.SHF/ALL,SPD.R/WRITE.RAB.SPC.RAB/01'
R[]_LA+D-1        'A7A/LA.+BYA/J.B7A/.BNX,ALU/A+B-1,SHF/ALL,SPC R/WRITE.RAB.SPD.RAB/01'
R[]_LA-D          'A7A/LA,B+BYA/J.B7A/.BNX,ALU/A-B.S-T/A.U,SPD.R/WRITE.RAB.SPC.RAB/01'
R[]_LA1B[]        'ANX.LA,.BX,-BX.A7A/B?.A.U,A-B.SHF/ALL.SPD.R/WRITE.RAB.SPD.R?1/01'
R[]_LA+K[+1       'ANX/LA,BNX/BYA.A7A/B2.ALU/A-B+'.B[#1],A.U'
R[]_LA+R[]        'ANA/LA.BNA/BNA.A7A/B2.ALU/A+B.ANX/ALL.SPD.R/WRITE.RAB.SPD.RA$/01'
R[]_LA+K[].RLOG   'ANA/LA.BNA/BNA.A7A/B2.ALU/A+B.RLOG.ST/ZERO,SHF/ALU.SPD R/WRITE.RAB.SPD.RAB/01'
R[]_LA+L[].RLOG   'ANA/LA.BNA/BNA.A7A/B2.ALU/A+B.RLOG.ST/ZERO,SHF/ALU,SPC.R/WRITE.RAB.SPD.RAB/01'
R[]_LA+L[].RLOG   'ANA/LA.BNA/BNA.A7A/B?.ALU/A+B.RLOG.ST/ZERO,SHF/ALU,SPC.R/WRITE.RAB.SPD.RAB/R0'
R[]_LA-K[].RLOG   'ANA/LA.BNA/BNA.A7A/B?.ALU/A-B.RLOG.ST/ZERO,SHF/ALU,SPC.R/WRITE.RAB.SPD.RAB/R0'
R[]_LA+D          'ANA/LA.BNA/BNA.A7A/B?.ALU/A+B,SHF/ALL,SPC.R/WRITE.RAB.SPD.RAB/01'
R[]_LA1?A+1[]     'ANA/LA.BNX/BNX.A7A/B?.A.U/A-B+'.B[#1]_A.U'
R[]_LA1?A+1[]     'ALU/A.B  .ANX/ A,BNX/BNX.SPD.R/WRITE.RAB.SPD.RA?/01,SHF/ALL'
R[]_LA.OR.D       'ANA/LA.BAN/Q.BNA/B?A.+ BYA/ALU/OP/DT,SHF/ALL.SPC.R/WRITE.RAB SPC.RAB/01'
R[]_LA.CRNTY.ANDM       'ANA/LA.BYA/BAS.ALU/OP/DT,SHF/ALL.SPC.R/WRITE.RAB SPC.RAB/01'
R[]_LA+D          'ANA/LA.BAN/Q.BNA/B?A.+ BYA/A.U/A-B,SHF/ALL SPD R/WRITE.RAB,SPD.RAB/01'
R[]_LA-D          'ANA,LA.BAN/Q.BNA/B?A.+ BYA/A.U/A-B,SHF/ALL SPD R/WRITE.RAB,SPD.RAB/01'
R[]_LB            'BNA/LA.A.U,B.S-T/A.U,SPC R/WRITE.RAB.SPD.RAB/01'
R[]_LC            'BA+/ C.A.U/B.S+T/A.U,SPC R/WRITE.RAB.SPD.RAB/01'
```

MICRODIAGNOSTICS CONTROL ROM FIELD DEFINITIONS

```
R[]_LC.RIGHT       "BMX/LC,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_NOT.C          "AMX/RAYA,DXT/DT/LONG,ALU/NOTA,R[#1]_ALU"
R[]_NOT.D          "RAYX/D,AMX/RAYA,ALU/NOTA,R[#1]_ALU"
R[]_NOT.MASK       "BMX/MASK,AMX/RAYX,DXT/DT/LONG,ALU/ORNOT,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_NOT.Q          "RAYX/Q,AMX/RAYA,ALU/NOTA,R[#1]_ALU"
R[]_PACK.FP        "BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_Q... THRU R[]_RLUC...
R[]_Q              "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU"
R[]_Q+1            "ALU_PAQ+1,R[]_ALU"
R[]_Q+B            "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/A+B+1,BYX/BMX,BMX/..,AMX/RAMX,RAMX/Q,SHF/ALU"
R[]_Q-C            "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAMX,BMX/LC,B/X/RBYX,ALU/A-B,SHF/ALU"
R[]_Q-C-1          "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/A-B-1,AYA/RAMX,RAYA/Q,BYX/RBYX,BMX/D,SHF/ALU"
R[]_DIX[]          "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAYX,BYX/RYX,BMX/#2,ALU/A+B,SHF/ALU"
R[]_D-R[]          "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAYX,BYX/RYX,BMX/#2,ALU/A-B,SHF/ALU"
R[]_Q.R[].RLOG     "RAMX/Q,AYA/RAMX,BYX/RYX,BMX/#2,ALU/A-B,RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_Q+LB           "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/A+B,AMX/RAMX,BMX/LB,RAMX/Q,SHF/ALU"
R[]_Q+LC           "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAYA,BYX/LC,ALU/A+B,SHF/ALU"
R[]_Q-LC           "SPO.R/WRITE.RAB,SPO.RAB/#1,RAMX/Q,AMX/RAYA,BYX/LC,ALU/A-B,SHF/ALU"
R[]_D.AND.R[]      "ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/#1,AMX/RAMX,RAYX/Q,BMX/#2,BMX/#2"
R[]_D.ANDNOT.R[]             "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/ANDNOT,AYX/RAMX,RAMX/Q,BMX/RMX,BMX/#2,SHF/ALU"
R[]_D.OR.D         "SPO.R/WRITE.RAB,SPO.RAB/#1,ALU/OR,AMX/RAMX,RAMX/Q,BMX/RBYX,RBMX/D,SHF/ALU"
R[]_Q.ORNOT.R[]    "SPO.R/WRITE.RAB,SPO.RAB/#1,RAYX/Q,AMX/RAMX,BMX/RMX,BMX/#2,ALU/ORNOT,SHF/ALU"
R[]_Q.RIGHT.1      "ALU_Q.SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_RLUC.RIGHT.1            "BMX/D,RYX/READ.R.LC,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/#1"
RC[]_D... THRU RC[]_D...

RC[]_0             "AMX/RAMX,DXT/DT/LONG,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0-D           "AMX/RAMX,DXT/DT/LONG,RBMX/D,BMX/RBYX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0+R[]+1       "AMX/RAMX,DXT/DT/LONG,BMX/#2,BMX/RMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0+LC+1        "AMX/RAMX,DXT/DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0+MASK+1      "AMX/RAMX,DXT/DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_0+MASK+1.RIGHT2     "AMX/RAYX,DXT/DT/LONG,BMX/MASK,ALU/A+B+1,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_ALU           "SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_ALU.LEFT      "SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_ALU.LEFT2     "SPO.R/WRITE.RC,SPO.RC/#1,SHF/ALU,BY.DT/LONG"
RC[]_ALU.LEFT3     "SPO.R/WRITE.RC,SPO.RC/#1,SHF/LEFT3"
RC[]_ALU.RIGHT     "SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_D             "RAYX/D,AMX/RAYX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_D.CAT[]       "RAYX/D,AMX/RAYA,DXT/DT/#2,ALU/[illegible],SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_D.AND.R[]     "RAYX/D,AMX/RAYA,BMX/RMX,BMX/#2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
```

```
RC[]_D.AND.MASK *RAYX/J,AMX/*AYX,BMX/MASK,ALU/AND,SHF/ALU,SPD.R/WRITE,RC,S-J.RC/#1*
RC[]_D.ANDNOT_Q *RAYX/J,AMX/*AYX,BMX/Q,BMX/*BYX,ALU/ANDNOT,SHF/ALU,SPD.R/WRITE,RC,S-J.RC/#1*
RC[]_D[6]       *RBYX/J,BMX/RBYX,A-1/J,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_D.CTX      *RAYX/J,AMX/RAYX,ALU/J,SHF/ALU,DT/DT/LSH,DEP,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_D+K[]      *RAYX/J,AMX/RAYX,BYX/KNX,KN/#2,ALU/A+B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_D-K[]      *RAYX/J,AMX/RAYX,BYX/KNX,KN/#2,ALU/A-B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_D._EFT     *RAYX/J,AMX/RAYX,A-1/J,SHF/LEFT,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_D.LEFT3    *RAYX/J,AMX/RAYX,A-1/J,SHF/LEFT3,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_D.OR.K[]   *RAYX/J,AMX/RAYX,KYX/#2,BMX/KNX,KN/OR,B47/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_D.SA.9     *RAYX/J,AMX/RAYX,BBMX/Q,BMX/BYX,ALU/OR,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_D.SMSGT.K[]        SPD.RC/#1,SPD.R/WRITE,RC,ALU/COMPUT,AMX/RAMX,RAMX/D.BYX/KMX,KNX/#2,SHF/ALU*
RC[]_D.SX+[]    *RAYX/J,AMX/RAYX,SX+/_D*/#2,ALU/A,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
;RC[]_A... THRU RC[]=A_...

RC[]_K[]        *AMX/#2,BMX/KMX,ALU/B,SHF/ALL,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_K[]+1      *AMX/RAMX,DXT,DT/LSHD,MX/#2,BMX/AYX,ALU/A+1,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_K[].LEFT2  *AMX/#2,BMX/KMX,ALU/B,SHF/ALL,DT,DT/LSHD,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_K[].RIGHT2 *KMX/#2,BMX/KMX,ALU/B,SHF/RIGHT2,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_L4         *AMX/L4,ALU/A,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_L4.AND.K[]         ALU_L4.AND.K[#2  FC/#1 _ALL*
RC[]_L4.L/R     *AMX/L4,ALU/A,SHF/ALU,DT,DT/LSHD,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_L4-K[]     *AMX/L4,KPX/#2,BMX/KYX,ALU/A-B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_L5         *BMX/L5,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_L5.LEFT    *BMX/L5,ALU/B,SHF/LEFT,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_L5         *BMX/L5,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_NOT.Q      *RAMX/Q,BMX/RAYX,ALU/NOTA,RC/#1*  ALU*
RC[]_PACK.FP    *BMX/PACKED,FL,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_SC         *BMX/SC,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q          *RAMX/Q,AMX/RAYX,ALU/B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q.DXT      *RAMX/Q,AMX/RAYX,DXT,DT/#2,ALU/A,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q.AND.K[]  *RAMX/Q,AMX/RAYX,BYX/KMX,KNX/#2,ALU/AND,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q.ANDNOT.K[]       RAYX/J,KYX/RAMX,BMX/KMX,+YX/#2,ALU/ANDNOT,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q+1        *ALU_Q+1*,RC[#]_ALU*
RC[]_Q+K[]      *RAYX/J,AMX/*AYX,BYX/KMX,KC+/#2,ALL/A+B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q-K[]      *RAYX/J,AMX/RAYX,BYX/KMX,KNX/#2,ALL/A-B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q._EFT     *RAYX/J,AMX/RAYX,A-1/A,SHF/LEFT,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q._EFT3    *RAYX/J,AMX/RAYX,A-1/A,SHF/LEFT3,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q+SSK+1    *RAYX/J,AMX/RAYX,BMX/MASK,ALL/A+B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q+FC       *RAYX/J,AMX/RAYX,BYX/FC,ALU/A+B,SHF/ALU,SPD.R/WRITE,RC,SPD.RC/#1*
RC[]_Q1PC1      *RAYX/J,AMX/*AYX,BYX/*Q,ALU/A+B*,SHF/ALU,SPD.R/WRITE,RC,S-J.RC/#1*
```

```
RE[]_C.RIGHT     "RAMX/D,4YX/RAMX,ALL/4,SHF/RIGHT,SPD.P/WRITE.RC,SPD.RC/P1"
RE[]_S.SYTL]     "RAMX/D,4YX/RAMX,SXT.DI/RP,ALL/4,SHF/ALL,SPD.R/WRITE.RC,SPD.RC/P1"
RC[]_RLSC.RIGHT "RAX.B,MRC/READ.RLSC,A 1/B,SHF/RIGHT,SPC R/WRITE.RC,SPD.RC/P1"
RC[]$VA_C!O      "RAMX/D,4YX/RAMX,PSY/C B/X/PRM3,ALU/A+B.VAR/LOAD,SHF/ALL.SPD.R/WRITE.RC,SPD.RC/P1"
RI[SC]_ALL       "SHF/ALL,SPD/WRITE.RC SC"
:RIDST]_... THRU R_16M3 ...
<[DST]_ALL       "SHF,A U,SPD.AC/WRITE.RAB,SPD.ACN:1/DST.DST"
R[DST]_D         "RAMX/D,4YX/RAMX,ALL/4,SHF/ALL,SPD.AC/WRITE.RAB,SPD.ACN:1/DST.DST"
P[DST]_D.SAT].RIGHT      "RAMX/D,AMX/-4YX.SX .D/.D/ ,D.D/A.SHF/RIGHT,SPD.AC/WRITE.RAB,STD.ACN:1/DST.DST"
<[PRN]_ALL       "SHF/ALL,SPD.AC,WRITE.RAB,SPD.ACN/PRN"
<[PRN]_D         "RAMX/D,4YX/RAMX,ALL/4,SHF/ALL.SPD.AC/WRITE.-AB.SPD.ACN/PRN"
A[PRN]_D.CR.C    "RAMX/D,4YX/RAMX,SBY.A/C B'A,4BMX,ALU/D3,P:PRN] ALU"
P[PRN]_D._O      "RAMX/D,4YX/RAMX,SBYA/C,4YX/MEM,ALU/M1,:PRN] ALU"
P[PRN]_D+R[].RLSC  D4YX/D,AMX/-4YX.4MX/M1,DM1/4MX,ALL/4+%.R_O3.DT/.ONG R:PRN]_ALU"
R.PRN]_D-R[].RLSC  D4YX/D,AMX/-4YX.4MX/M1,D%1/4MX,ALL/4-%.R_O3.DT/.ONG R.PRN]_ALU"
S[PRN]_D[]       MYX/M1 DXX/4YX,A1 /S.SHF/ALL,SPD.AC/WRITE.RAB,SPD.ACN/PRN"
-:[PRN] [4:0][].RLSC      "4YX/LA.XM%/M1,BMX/4YX,ALL/4+%.R_O3,D'/LONG.R:PRN] ALU
R[PRN]_[6-R[].RLSC        "4YX/LA.BMX/M1,BMX/M4X,ALL/4-0.R_O3,D'/LONG.R[PRN] ALU
R[PRN]_[4._MASK  AMX/LA.BMX/MASK,ALU/M1,SHF/ALL.SPD.AC/WRITE.RAB.SPD.ACN/PRN"
R[PRN]_[4+0      AMX/LA.-BYA/D,BYA/.BMX,ALL/A+B.SHF/ALU,SPD.AC/WRITE.RAB,SPD.ACN/PRN"
R[PRN]_FASK FP   BMX/PASK.EP,ALL/4,SHF/ALL.SPD.AC/WRITE.RAB.SPD.ACN/PRN"
R[PRN]_D         "RAYX/D,AMX/-4YX.ALU/4,SHF/ALL,SPD.AC/WRITE.RAB,SPD.ACN/PRN
R[PRN-1]_D       "RAYX/D,AMX/-4YX.ALU/3,SHF/ALL,SPD.AC/WRITE.RAB,SPD.ACN/PRN+1"
R[PRN-1]_D.D3.0  "RAYX/D,AMX/-4YX.BMX.D/.BMX/PSYX.A 1/CP,SHF/ALL,SPD.AC/WRITE.RAB.SPD.ACN/PRN+1"
D[PRN-1]_LD      "BMX/LD,ALU/1,SHF/ALL.SPD.AC/WRITE.RAB.SPD.ACN/PRN+1"
R[PRN-1]_D       "R4YX.D,MMX/R4YX,A 1/B,SHF/ALL,SPD.AC/WRITE.RAB,SPD.ACN/PRN+1"
R[SC]_A.C        "SHF/ALL,SPD.AC/WRITE.RAB.SHF ACN/SC"
R[SC]_D          "RAMX/D,AMX/RAYX,ALL/A,SHF/ALL,SPD.AC/WRITE.RAB,SPD.ACN/SC"
R[SC]_D+3        "AMX/LA.BMX/D,PMX/+B.ALU/A+B,SHF/ALL.SPD.AC/WRITE.RAB,SPD.ACN/SC"
R[SC]_D4-3       "AMX/LA.MBMX/D,BMX/-B/A,ALU/A+2,SHF/ALL.SPD.AC/WRITE.RAB,SPD.ACN/SC"
R[SC]_LD                 ALU_LD, :[SC] ALL"
R[SC]_D          "AMX/Q,AYX/BMX,ALL/A-S.F/A U,SPD.AC/WRITE.RAB,SPD.ACN/SC"
R[SPTTT]_LD      "DM*/.D,A.1/B,%.*/AL.1,SPT.RC/WRITE.RAB,SPC.ACN/SP:.1"
R[SP1+1]_D       "RAMX/D,4YX/RAMX,ALL/A,SHF/ALL,SPD.AC/WRITE.RAB,SPD.ACN/SP:+1"
R[SP1]_LD        "SP:,ALU,B.D.ACN.WRITE.RAB,SPD.ACN/SP1.SP1"
R[SP1]_D         "RAMX/D,4YX/RAMX,ALL/A,%.F/ALL.SPD.AC/WRITE.RAB,SPD.ACN/SP1.SP1"
A[SP1]_A[]       "4YX/X  DYX/4YX,ALU/B,SHF/ALL,SPD.AC/WRITE.RAB.SPD.ACN/SP1.SP1"
A[SP1]_PACK.FP   "SYX/PAC-EC.FL,ALU/B.SHF/ALL,SPD.AC/WRITE.RAB.SPD.ACN/SP1.SP1"
R[SP1]_D         "RAMX/D,4YX/RAMX,ALL/A,SHF/ALL.SPD.AC/WRITE.RAB,SPD.ACN/SP1.SP1"
R[SPD]_ALL       "SHF/ALU,B.D.AC/WRITE.RAB.SPD.ACN:1/SPD.SPD"
```

MICRODIAGNOSTICS CONTROL ROM FIELD DEFINITIONS

```
R[SRC]_D             "RAVA/D,AMX/RAVA,ALU/A,SHF/ALU,SPD.AC/WRITE.RAB.SPD.ACNT'/S.[illegible]"
R[SRC]_R[RN]         "RAVA/D,SVX/RAMX,ALU/B,SHF/ALU,SPD.AC/WRITE.RAB.SPD.ACNT'/[illegible]"
R[SRC]_D-K[].RLOG    "RAVX/D,AMX/RAVA,AMX/#1,BMX/KMX,ALU/A+B,RLOG,DT/WORD,R[SRC]_ALU"
R[SRC]_D-K[].RLOG    "RAVX/D,AMX/RAVA,AMX/#1,BMX/KMX,ALU/A-B,RLOG,DT/WORD,R[SRC]_ALU"
R[SRC]_LC            "SVX/LC,ALU/B,R[SRC]_ALU"
R[SRC]_Q             "RAVA/Q,AMX/RAMX,ALU/A,SHF/ALU,SPD.AC/WRITE.RAB.SPD.ACNT'/S.[illegible]"
R[SRC]_[illegible]_ALU  "SHF/ALU,SPD.AC/WRITE.RAB,SPD.ACNT/SRC.OR.[illegible]"
R[SRC]_[illegible]_D[B]  "RMX/D,SVX/RBMX,ALU/B,SHF/ALU,SPD.AC/WRITE.RAB,SPD.ACNT/SRC.OR.[illegible]"
R[]AVA_LA+K[]        "SVX/LA,AVX/#0,BVX/KMX,ALU/A+B,VAR/LOAD,SHF/ALU,SPD.R/WRITE.RAB,SPD.RAB/#1"
R[]AVA_LA-K[]        "SVX/LA,AVX/#0,BVX/KMX,ALU/A-B,VAR/LOAD,SHF/ALU,SPD.R/WRITE.RAB,SPD.RAB/#1"
R[]AVA_LA-K[].RLOG        "AMX/LA,BMX/#0,SVX/SVX,ALU/A-B,RLOG,DT/LONG,VAR/LOAD,SHF/ALU,SPD.R/WRITE,RAB,SPD.RAB/#1"
R[]AVA_Q-K[]         "RAMX/Q,AVX/RANX,AVX/#0,BVX/KMX,ALU/A-B,VAR/LOAD,SPD.R/WRITE.RAB.SPD.RAB/#1"
;SC ...

SC_0(A)              "ASX/RANX,CXT,DT/LONG,SBVX/AVX,EXP,EALU/A,SMX/EALU,SCK/LOAD"
SC_D-K[]             "RAVX/KMX,PMX/#1,AVX/AMX,CAT,DT/LONG,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_ALU               "SMX/ALU,SCK/LOAD"
SC_ALU(EXP)          "SMX/ALU,EXP,SCK/LOAD"
SC_D                 "RANX/D,AMX/RAVX,ALU/A,SVX/ALU,SCK/LOAD"
SC_D.CXT[]-K[]       "RAVX/D,AMX/RAVX,CXT,DT/#1,AMX/#1,BMX/KMX,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_D.CXT[].XOR.K[]           "RAVX/D,AVX/RANX,CXT,DT/#1,SMX/AVA,AVA/#1,ALU/XOR,SC_ALU"
SC_D.AND.K[]         "RAVX/D,AMX/RAVA,KMX/#1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD"
SC_D(EXP)            "RAVX/D,AMX/RAVA,ALU/A,SVA/ALU,EXP,SCK/LOAD"
SC_D(EXP)(A)         "RAVX/D,AMX/RAVA,BMX/AVA,EXT,EALU/B,SMX/EALU,SCK/LOAD"
SC_D(EXP)(B)         "RBVA/D,SMX/RBVA,ALU/B,SMX/ALU,EXP,SCK/LOAD"
SC_D-K[]             "RAVA/D,AMX/RAVA,KMX/#1,SMX/AMX,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_D.OR.K[]          "RAVA/D,AMX/RAVA,KMX/#1,BMX/KMX,ALU/OR,SMX/ALU,SCK/LOAD"
SC_D.SXT[]           "RAVA/D,AMX/RAVA,SXT,DT/#1,ALU/A,SVA/ALU,SCK/LOAD"
SC_EALU              "SMX/EALU,SCK/LOAD"
SC_FE                "SVX/FE,SCK/LOAD"
SC_NABS(SC-FE)       "FAV/FE,EALU/NABS.A-B,SMX/EALU,SCK/LOAD"
SC_K[]               "KVX/#1,SVX/KMX,EALU/B,SMX/EALU,SCK/LOAD"
SC_K[].ALU           "KVX/#1,SVX/AVX,ALU/B,SMX/ALU,SCK/LOAD"
SC_LA                "SVA/LA,ALU/A,SMX/ALU,SCK/LOAD"
SC_LA.AND.K[]        "AVX/LA,KVX/#1,BVX/KMX,ALU/AND,SVX/ALU,SCK/LOAD"
SC_LC(EXP)           "SVX/LC,ALU/B,SVX/ALU,EXP,SCK/LOAD"
SC_PSLADDR           "SVX/EALU,SBVX/AVA,SCK/LOAD,AVX/[illegible],EALU/B"
SC_Q                 "RAMX/Q,AVX/RANX,ALU/A,SMX/ALU,SCK/LOAD"
SC_Q.AND.K[]         "RAMX/Q,AVX/RANX,BVX/KMX,KMX/#1,ALU/AND,SVX/ALU,SCK/LOAD"
SC_Q(EXP)            "RAMX/Q,AVX/RANX,SBVX/AMX,EXP,EALU/B,SVX/EALU,SCK/LOAD"
```

```
SC_Q(EXP)(6)      'MBYA/Q,BNX/-BYA.ALU/S.SMA/ALU.EXP.SCK/LOAD'
SC_Q-A[]          'MAYA/Q,ANX/-AYA,BMA/MNX.AYA/01.ALU/A-B.SMA/ALU,SCK/LOAD'
SC_Q-A[]          'MAYA/Q,AMX/-AYA,BMA/MYA.AYA/01.ALU/A-B.SMA/ALU,SCK/LOAD'
SC_Q.OR.A[]       'MAYA/Q,ANX/PAYA,BMA/ZMX.AYA/01.ALU/OR.SMA/ALU,SCK/LOAD'
SC_Q.AXY[]        'MAYA/Q,ANX/PAY*.5*T,DT/01,ALU/A,SMX/ALU,SCK/LOAD'
SC_R[]            'SPC.R/.LOAD.LAB.SPD.RAB/01,ANX/16,ALU/A,SNX/ALU,SCK/LOAD'
SC_RC[]           'SPC.R/.LOAD.LB.SPD.PC/01.BYA/LB.ALU/B,SYA/ALU,SCK/.LOAD'
SC_R[].AND.K[]    'ALL/ANE.AYA/LA.SPD.R/LOAD..AB.SPC.RAS/01.BYX/KMA.BNX/02.SYX/ALU.SCK/LOAD'
SC_R[](EXP)       'SPC.R/.LOAD.LB.SPD.RAB/01,BNX/16,ALU/A,SNX/ALU.EXP.SCK/.LOAD'
SC_RC[].(EXP)     'SPC.R/.LOAD.LB.SPD.PC/01.BYA/LB.ALU/B,SYA/ALU.EXP,SCK/LOAD'
SC_SC+1           'EALU/A+1,SMX/EALU.SCA/.LOAD'
SC_SC.ANDNOT.FE   'EBYA/FE.EALU/ANDNOT.SYA.EALU,SCK/LOAD'
SC_SC.ANDNOT.K[]  'KYA/01.EBNX/AYA.EALU/ANDNOT,SNX/EALU,SCK/LOAD'
SC_SC+FE          'EBYA/FE.EALU/A-B.SYA/EALU,SCK/LOAD'
SC_SC-FE          'EBYA/FE.EALU/A-B.SYA/EALU,SCK/LOAD'
SC_SC+K[]         'KMA/01.EBYA/KMA.EALU/A+B.SMA/EALU.SCK/LOAD'
SC_SC-K[]         'KMX/01.EBYA/KMA.EALU/A-B.SMA/EALU.SCK/LOAD'
SC_SC.OR.K[]      'KMX/01.EBYA/KMA.EALU/OR.SYA/EALU,SCK/LOAD'
SC_SC+SHF.VAL     'EBYA/SHF.VAL,'ALU/A+B,ZNX/EALU.SCK/LOAD'
SC_SHF.VAL        'EBYA/SHF.VAL,'ALU/B,SYA/EALU,SCK/LOAD'
SC_STATE          'EALU/A.MSC/LOAD.STATE,SNX/EALU..SCA/.D4C'
SC_STATE.ANDNOT.K[]       'EALU/ANDNOT.EMX/KMX,MSC/LOAD.STATE.SMX/EALU,SCK/.LOAD,KYA/01'
SCSSTATE_STATE-K[](EXP)  'LAB_R[01].AMX/L4.EBNX/AYA.EXP.MSC/LOAD.STATE.EALU/A-B.SYA/EALU.SCK/LOAD'
;SD_... (NNU VA_...

SS_NOT.SS         'SSA/NOT.SS'
SS_SS             'SSA/SS.FREM.SS'
SS_CASD_S         'SSA/S'R 'SS'SS'
SS_ALUIS          'SSA/LOAD.SS'
SS_SO             'SSA/SS.FREM.SO'
SS_SS.XOR.ALUISSSD_ALUIS          'SSA/SS.XOR ALU'

STATE,0(A)        'SYA'RAM'.B*T,DT/LONG,BBNX/AMA.EXP.BALU/B,MSC/LOAD.STATE'
STATE_AMX,EAP     'EBYA/AP'.E*P,TA'U/B.MSC/LOAD.STATE'
STATE_D(EXP)      'MAYA/D.AYA/MMX,EBYA/AMY.EXP,'ALU/B,MSC/LOAD.STATE'
STATE_FE          'EBYA/FE.EALU/B.YSL/.LOAD STATE'
STATE_K[]         'KYA/01.EBYA/KMA,EALU/B,MSC/LOAD.STATE'
STATE_Q(EXP)      'MAY*/Q,YYX/RMA,EBYA/AMA.EXP,EALU/B,MSC/LOAD.STATE'
STATE_SO.VIA.KMX          'MSC/LOAD.STATE.EALU/B,EMA/KMX,MNX/AD'
STATE_STATE-1     'EALU/A-1,MSC/LOAD.STATE'
STATE_STATE.ANDNOT.FE     'EBNX/FE.EALU/ANDNOT.YSC/LOAD.STATE'
```

```
STATE_STATE.ANDNOT.K[]   "AVA/S?.ESMX/AVS.EALU/ANDNOT,MSC/LOAD.STATE"
STATE_STATE+FE   "EBVA/FE.EALU/A+B,MSC/LOAD.STATE"
STATE_STATE-FE   "EBVA/FE.EALU/A-B,MSC/LOAD.STATE"
STATE_STATE+K[]  "KMX/#1,EBVA/KMX,EALU/A+B,MSC/LOAD.STATE"
STATE_STATE-K[]  "KMX/#1,EBVA/KMX,EALU/A-B,MSC/LOAD.STATE"
STATE_STATE.OR.FE        "EALU/OR,EBMX/FE,MSC/LOAD.STATE"
STATE_STATE.OR.K[]       "KMX/#1,EBMX/KMX,EALU/OR,MSC/LOAD.STATE"
;SKPC STATES
STATE_SK.LONG            "STATE_K[.4]"
STATE_STATE.AN.SKPLONG   "STATE_STATE.ANDNOT.K[.4]"
;EDITPC STATES
STATE_FIRST              "STATE_K[ZERO]"
STATE_PREDEC             "STATE_K[.80]"
STATE_STATE.AN.STEP      "STATE_STATE.ANDNOT.K[.3F]"
STATE_STATE.AN.STEP4     "STATE_STATE.ANDNOT.K[.7F]"
STATE_STATE.AN.DESTDN    "STATE_STATE.ANDNOT.K[.5]"
STATE_STATE.AN.NOTPREDEC       "STATE_STATE.ANDNOT.K[.7F]"
STATE_STATE.AN.PREDECZERO      "STATE_STATE.ANDNOT.K[.C0]"
STATE_STATE.OR.ADDINP    "STATE_STATE.OR.K[.8]"
STATE_STATE.OR.DEST      "STATE_STATE.OR.K[.4]"
STATE_STATE.OR.DESTDBL   "STATE_STATE.OR.K[.3]"
STATE_STATE.OR.FILL      "STATE_STATE.OR.K[.7]"
STATE_STATE.OR.FLOAT     "STATE_STATE.OR.K[.80]"
STATE_STATE.OR.MOVE      "STATE_STATE.OR.K[.50]"
STATE_STATE.OR.PATT1     "STATE_STATE.OR.K[.1]"
STATE_STATE.OR.PATT2     "STATE_STATE.OR.K[.2]"
;MATCHC STATES
STATE_INNERPOS   "STATE_K[.1]"
STATE_INNERSRC   "STATE_K[.3]"
STATE_OUTER      "STATE_K[ZERO]"

SWAPD            "DK/BYTE.SWAP"

VA_ALU           "VAK/LOAD"
VA_D             "RAVX/D,AMX/RAMX,ALU/A,VAK/LOAD"
VA_D.SXT[]+Q      "RAVA/D,AMX/RAMX,CALT.DT/#1,BMX/RBVA,ALU/A+B,VAK/LOAD"
VA_D.ANDNOT.K[]   "RAVA/D,AVX/RAMX,BMX/KVA.KVA/#1.ALU/ANDNOT,VAK/LOAD"
VA_D-K[]          "RAVA/D,AVX/RAMX,BMX/#1.BVX/KVA.ALU/A-B,VAK/LOAD"
VA_D+LC          "RAVA/D.AVX/RAMX,BMX/LC.ALU/A+B,VAK/LOAD"
VA_D+Q           "RAVA/D.AVX/RAMX,RBVA/Q.BVX/RBMX,ALU/A+B,VAK/LOAD"
```

```
VA_A[]              *AMX/#1,BMX/BMX,ALU=B,VAK/LOAD*
VA_LA               *AMX/A,__ I/A,VAK/LOAD*
VA_LA.AND.LC        *AMX/A,BMX/C,ALU/AND,VAK,LLRJ
VA_L5.ANDNOT.A[]            AMX/A,BMX/C,ALU/ANDNOT,VAK/LOAD*
VA_LA#3             *AMX/A,BBMX/D,BMX/#3[],ALU=B,VAK/LOAD*
VA_LA=[]            *AMX/A,BMX/D,BMX/#3[],ALU=B,VAK/LOAD*
VA_LA+4[]           *AMX/A,BMX/BMX,BMX/#1,ALU/A+B,VAK/LOAD*
VA_LA-4[]           *AMX/A,BMX/BMX,BMX/#1,ALU/A-B,VAK/LOAD*
VA_L4+X[]-1         *AMX/A,BMX/BMX,BMX/#1,ALU/A+B-1,VAK/LOAD*
VA_L4-X[]-1         *AMX/A,BMX/BMX,BMX/#1,ALU/A-B-1,VAK/LOAD*
VA_L4[]PC           *AMX/A,BMX/PC,ALU/A+B,VAK/LOAD*
VA_L4+Q             *AMX/A,BBMX/Q,BMX=.BMX,ALU/A+B,VAK/LOAD*
VA_L4-Q             *VAK/LOAD,ALU/A-B,A=A,BMX/BBMX,BMX/Q,SHF/ALU*
VA_L8+D.OAT         *AMX/LB,ALU/A+B,AMX/A/A.OAT,LC/BYTE,VAK/LOAD*
VA_'X'              BMX.PC,ALU/B,VAK/LOAD
VA_Q                BAMX/Q,ALU/AMA,ALU/A,VAK/LOAD*
VA_Q.AND.NOT.X[]    *BAMX/Q,AMX/BAMX,BMX/#1,"NOT"BMX,ALU/AND.NOT,VAK/LOAD*
VA_Q=D              *VAK/LOAD,ALU/A=B,BMX/BAMX,BMX/Q,BBMX/D,SHF/ALU*
VA_Q=A[]            *BAMX/Q,AMX/BAMX,BMX/#1,BMX/BMX,ALU/A+B,VAK/LOAD*
VA_Q=A[]            *BAMX/Q,AMX/BAMX,BMX/#1,BMX/BMX,ALU/A-B,VAK/LOAD*
VA_Q+LC             *BAMX/Q,AMX/BAMX,BMX/LC,ALU/A+B,VAK/LOAD*
VA_Q+LB             *BAMX/Q,AMX/BAMX,BMX/LB,ALU/A+B,VAK/LOAD*
VA_Q+LB,PC          *BAMX/Q,AMX/BAMX,BMX/PC,DA.LB.ALU/A+B,VAK/LOAD*
VA_Q+PC             *BAMX/Q,AMX/BAMX,BMX/PC,ALU/A+B,VAK/LOAD*
VA_B[]              *SPD.R/LOAD.LAB,SPD.PAR/#1,ANX/LA,ALU/A,VAK/LOAD*
VA_BE[]             *SPD.R/LOAD.LB.SPD.RC/S1.BMX/LB.ALU/B,VAK/LOAD*
VA_VA+4             *PCK/VA+4
        "NON-EXECUTABLE" FUNCTIONS*

B.FORK              *LAB_BIGFI[],QK/CB,CIR.IR.COND,PC_PC=N,SUB/SPECIAL/B.FORK*
BYTE                'BY/BYTE*
CACHE.INVALIDATE            *NOT/INVALIDATE,VAK/NOP*
CALL                *SUB/CALL*
CAL ()              *CALL,J/$1*
C.FORK              *SUB/SPECIAL/C.FORK*
CHK.DOB.BJOB        *VSD/CHK.DOB.ADDR*
CHK.FLT.OPR         *WSD/CHK.FLT.OPR*
CLK.USED            *DCA/LOAD.UBDC*
CLR.FPD             *NOT/CLR.FPD*
CLR.ABO=1           *JSD/CLR.0 1.(EX/ISTR*
```

```
CLR.IBC=3       'IBC/0,0,0-3'           ;DISCARD -11 (WITH A OPERAND
CLR.IBS=3       'IBC/0,0,2-3'           ;11 MODE DISCARD (STRIKE OPERAND
CLR.IBZ=5       'IBC/0,0,1-5,COND'      ;2ND PART OF S/D IMMEDIATE
CLR.IB.2ND      'IBC/0,0,1-5,COND'
CLR.IB.DPC      'IBC/0,0,0,EX/JS-3'
CLR.IB.SPEC     'IBC/0,0,1'
CLR.NEXT.ESE    'MSC/CA,NCST.EXH'
CLR.SEXSS       'SAM/CA,SD-SS'
EXCEPT.AGN      'FILBACK'
FLUSH.IB        'IBC/IJSP.VAR/LOAD.TEK:1511'
INHIBIT.IB      'NOP/PEN.NOP'
IN.INT.HOK      'IBC/PCSE'
IN.INT.STROBE   'IBC/PSTB'
IRD             'IBC00.2LR.IMEC,L-JJ.SUS/SPEC,A4.FOHH'
IRTD            'LA_RISP2&IF_RISPIJ,LA44_LB/SC_PLUICXP),PE (&IEXP),SS_HLJ15'
ESH             'PSS/.JB.04/FO,WCT/SLJU4.IS.RESG,IBC/SIR.1-5.COND,/CK/PC-1'
EOD,-1          'L4_RLJSIIA&_RISRC1,P LE,/C,44A/LOAD.S_IB-DATA.SC_K[-10],PSC/(P2+N,MSC/(P2),SUB/SPEC,W/OP1'
.OAD.IB         'YSS/NTF.WCT/SEJ3.YPYAPC'
LOAD.IB.11      'YSS/NTF.WCT/SLAJ.YSAPC'
LUNG            'BT,LOWS'
SELY.DONE       'PSP/SKTSTL.ACM/FULY.DCHE'
RTTURN(1        'SJE/RET.J/04'
RTTURN0         'SJE/RET.J/0'
RTTURN1         'SJF/RET.J/1'
RETURN2         'SUM/RET.J/2'
RETURN3         'SUB/RET.J/3'
RETURN8         'SUB/RE-.J/8'
RETURN9         'SUB/RE-.J/9'
RETURNF         'SUB/RE-.J/F'
RETURN10        'SUB/RET.J/10'
RETURN12        'SUB/RET.J/12'
RETURN18        'SUB/RET.J/18'
RETURNIF        'SUB/RET.J/1F'
RETURN0C        'SUB/RET.J/0C'
RETURN24        'SUB/RET.J/24'
RETURN2C        'SUB/RET.J/2C'
RETURN3C        'SUB/RET.J/3C'
RETURN40        'SUB/RET.J/40'
RETURN50        'SUB/RET.J/50'
```

```
RET.RN:DE       'SUB/RET',0/10E'
SET.CC(INST     'CCK/INST,DEP,DT/INST DEP'
SET.CC(LONG)    'CCK/INST,DEP,D7/LONG'
GET.CC(ERR)      CCK/RDB'
SET.FPD          MSC/SET FPD'
SET.N.GND.2     'CCP,TST 2'
SET.NEST.ERR    'MSC/SET.NEST.ERR'
SET.NAZ         'CCK/N+Z_ALL'
SET.PSL.C(AMX)  'CCK/AC_AMAX'
SET.V           'CCK/SET.V'
START.IE        'ISC/SIRT'
STOP.IE         'ISC/STOP'
TEST.YB.ROMK    'NCT/TTST.ROMK.YAK/KOP'
TTST.YB.WOMK    'NCT/TEST.WOMK.YAK/KOP'
TRAP,ACC:]      'ECF/TRAP,ACC/WI'
WORD            'DT/WORD'
WRITE.DTST       JAC,RISFI,,OK/ID,CLR,IS,CCK3,PC,FCKR,SUB/SPEC,,/ARD'
        'Branch Error w Zero Offset'

ACCEL?          'BEN,ACCEL'
ACC.SYNC?       'BEN/ACCEL'         ;,W3/3'
AC-CW?          'BEN/CNTS-RUP'      ;,W2/2'
ALIGNED?        'BEN/IS.TEST'       ;,W7/17'
ALU?            'BEN/ALU'
ALU.49          'BEN/ALU'           ;,W4/07'
ALL1-01         'BEN/ALU1-0'
BOOGUS?         'BEN/BOOTSEL'       ;,W2/2'
CBI?            'CLK/CB'
CONSOLE.MODE?    CEN/PSL.MODE'      ;,W5/10'
D07             'BEN/D8-0'          ;,W4/0E'
D1[]?           'BEN/N,,'
D2?             'BEN/D8-0'          ;,W3/0B'
D2-09           'BEN/D8-0'          ;,W4/08'
D8?             'BEN/D8-0'          ;,W4/07'
D8-D7           'BEN/D8-0'
D8?1            'BEN/BTAGS'         ;,W3/8'
DATA.TYPE?      'BEN/DATA.TYPE'
D.E0?           'BEN,D.0-IEA'       ;,W4/0E'
D.E1?           'BEN,D.E-IEA'       ;,W4/0E'
D.E2?           'BEN,D.E-IEA'       ;,W4/0E'
```

MICRODIAGNOSTICS CONTROL ROM FIELD DEFINITIONS

```
[illegible]?                  [illegible]/DATA.TYPE"
[illegible]BY.EN?             "[illegible].BYTES"
[illegible].SEL?              SEN.SIGNS"          ;[illegible]     ;PREFERRED FORM
ALU?                          "SEN/EALU
[illegible].U.N?              "[illegible]EALU           ;.04/07"
[illegible].U.Z?              "[illegible]/EALU"         ;.04/05"
END.[illegible]?              "[illegible]/[illegible].LP1"
FED?                          "BEN/[illegible].JEF"      ;[illegible]/07
IB.TEST?                      "BEN/[illegible]"
INT?                          BEN/INTERRUPT"
INTERNAL T.RED?               [illegible] INTERRUPT"     ;.03/[illegible]
IRD?                          "[illegible]"              ;.03/0D"
[illegible]                   "[illegible]"
IR[illegible]?                "BEN/[illegible]"          ;[illegible]/5"
[illegible]                   "BEN/[illegible]"
LAST.REF?                     "BEN/LAST.[illegible]"
[illegible].LSB.LASTLV?       "BEN/RSL"                  ;[illegible]
MM[illegible]?                "BEN/[illegible]"
N[illegible]LEN?              "BEN.LAST.RATE            ;.04/0B"
PC.MODE?                      "BEN/PC.MODES
PS[illegible]?                "BEN/PSL.[illegible]"      ;.04/0E"
PS[illegible]?                "BEN/PSL.[illegible]"
PSL.MODE?                     "BEN/PSL.MODE"
PSL.[illegible]?              "BEN/PS[illegible]         ;[illegible]/7"
PSL.N?                        "BEN/PSL.[illegible]"      ;[illegible]/0E"
PSL.Z?                        "BEN/PSL.[illegible]"      ;.04/0D"
PTE.VALID?                    "BEN/TS.TEST"              ;.05/0F"
QLAJ?                         "BEN/DATA.TYPE"
QST?                          "BEN/SIGNS"                ;.08/3"
RQ[illegible].EMPTY?          "BEN/[illegible]"          ;.04/7"
RDR?                          "BEN/ROM"
SC?                           "BEN/SC"
SC.GT.[illegible]             "BEN/SC"
SC.N[illegible]?              "BEN/[illegible]"          ;.03/3
SIGNS?                        "BEN/SIGNS"
SRC.PC?                       "BEN/SRC.[illegible]"      ;COMP MODE, BEN ON SRC R = PC
S[illegible]?                 "BEN/EALU"                 ;.04/0E"
STATE0?                       "BEN/STATE3-0"             ;[illegible]/0E"
STATE1?                       "BEN/STATE3-0"             ;.04/0B"
STATE1-0?                     "BEN/STATE3-0              ;.04/0C"
STATE2?                       "BEN/STATE3-0"             ;.04/0B"
```

```
STATE07             BSR/STATE3-07    ;;4/07*
STATE3-07           BSR/STATE3-07
STATE07             *BSR/STATE7-07*
STATE07             *BSR/STATE7-07*
STATE07             *BSR/STATE7-07*
STATE07             *BSR/STATE7-07*
STATE07                              STATE7-07
STATE7-07           *STR/STATE/-07*
IR.TEST?            *TBR/TB.TEST*
VAR1?               *SBR/PS..VECTOR    ;;05/07*
VAR 1SB1            BSR/PS..VECTOR     ;;0/07*
Z?                  BSR/Z
ZONED?              *BSR/SPECIAL*      ;;;/**
ALEG_E              *ANX/RAMX,RA/X/E*
ALEG_LA             *ANX/LA*
ALEG_LAB            ANX/LA*
ALEG_Q              ANX/RAM*,RAMX/Q
ALU_C(0)            ORG/ZERO,ENX/HOT,ALL_B
ALU_C_EQ            ANX/RAMX,OUT_OUT/LOAD_ADDR/D_RTA/RAMX,ALU/A+B+1*
ALU_C_LOAD_Q        *RAMX/Q,RAM/RAMX,ENX/Q,TMX/TBUS,ALU/AND*
ALU_C_ORRMASK       *RAMX/Q,RAM/RAMX,RSA/RAMX_ALL_ZERO*
ALU_C[]_B           *RAMX/Q,RAM/RAMX,ENX/_B,ALU/B*
ALU_C[]_ONE         *RAMX/Q,RAM/RAMX,ENX/ONE*/RAMX,ALL/B;*
ALU_C[]0[*          *ANX/RAMX,RAMX/Q,LC_CL[#2],OUT/ C,ALU/B*
ALU_C[]1[*          *ANX/RAMX,RAMX/Q,LC_CL[#2],OUT/CB,ALU/B*
ALU_C=F             *RAMX/RAMX,RAM/Q,LA[_E[#1] ATX/LA,ALU/A-B*
ALU_LA=LC           *ATX/LA,ENX/LC,ALU/A=B*
ALU_LA_AND_LC       *ATX/LA,ENX/LC,ALU/AND*
ALU_LA[]C_]         *ATX/LA,ENX/RAM,ANX/#2,ALU/#1*
ALU_LA[]LC          *ATX/LA,ENX/LC,ALU/A*
ALU_MASK            *BTX/MASK,ALL/B
ALU_MASK+1          *ATX/RAMX,C+T,OUT/ CAR_ENX/MASK,ALL/A+1*
ALU_NOT_C[]         *ANX/RAM-,C+T,OUT/ ONE_ATX/*,BTX/RMX,ALU/OPNOT*
ALU_NOT.LA          ANX/LA,*LL_*NOTA
ALU_NOT.MASK        ANX/RAMX,C+T,OUT/_ONE_BTX/MASK,ALU/OPNOT*
ALU_NOT.Q           A*TX/Q,ANX/ RAMX,ALU/NOTA*
ALU_Q=B *RAMX/Q     ATX/RAMX,ENX/BUS,ALU/A+B*
ALU_Q_TC            *RAMX,SC,ANX/ATX,RSA/Q_ ATX/RAMX,ALL/A-B*
ALU_Q[]_B           *RAMX/Q,ANX,E_D*_BTX/_*_ALU/B*
ALU_Q[]A.B          *RAMX/Q,ALU/RAMX,BTX/_A,B,ALU/#1*
ALU_Q.AND.MASK      *RAMX/Q,RAM/_ATX,BTX/MASK,ALU/AND*
```

MICRODIAGNOSTICS CONTROL ROM FIELD DEFINITIONS

```
ALU_Q.OR.MASK       "RAVX/Q,AMX,[illegible]MX,BMX,MASK,ALU/OR"
ALU_Q.ORNOT.MASK             "RAMX/Q,AMX/RAMX,[illegible]/MASK,ALU/ORNOT"
ALU_Q.XOR.RC[]      "[illegible]RC[@1],[illegible]/RAMX,[illegible]/Q,[illegible]MX/LC,ALU/XOR"
ALU_Q.XOR.R[SC]     "LAB-R[SC],BYX/LR,AYX/RAMX,RAYX/Q,ALU/XOR"
ALU_Q[]K[]          "RAMX/Q,AMX,RAYX,[illegible],BYX/KMX,ALU/A"
ALU_Q[]+D           "RAMX/Q,A[illegible]/RAYX,B[illegible]/D,ALU/A"
ALU_Q[]RC[]         "AYX/RAMX,RAMX/Q,LC_RC[@2],BYX/LC,ALU/A"
ALU_Q[]R[]          "AMX/RAYX,RAMX/Q,LAB_R[@2],BMX/LA,ALU/A"
ALU_R[SC]           "SPO.R/LOAD,LA[illegible],[illegible]/SC,AMX/LA,ALU/A
ALU_R[]+S           "MSD/READ,[illegible],ALU/B
ALU_R[].AND.MASK             "SPO.R/LOAD,LAB,SPO.RAB/@1,AYX/LA,BYX/MASK,ALU/AND"
ALU_R[].OR.MASK     "SPO.R/LOAD,LAB,[illegible]/@1,AMX/LA,BMX/MASK,ALU/OR
ALU_R[].ORNOT.MASK           "SPO.R/LOAD,LAB,SPO.RAB/@1,AYX/LA,BMX/MASK,ALU/ORNOT"
ALU_R[]+R[]         "SPO.R/LOAD,LAB,SPO.RAB/@1,AMX/LA,AYX/@2,BYX/RMX,ALU/A+B"
ALU_R[]-R[]         "SPO.R/LOAD,LAB,SPO.RAB/@1,AYX/LA,AYX/@2,BYX/RMX,ALU/A-B"
ALU_SC              "AYX/SC,BMX/AMX,ALU/B"
B_RG.D              "BMX/RAYX,RBMX/Q"
BLEQ_LR             "BMX/LR"
BLEQ_LC             "BMX/LC
BLEQ_Q              "BMX/RBY*,RBMX/Q
BPSYNC              "ACF/SYNC"
DELAY               "CALL[DELAY]"
D_R+K[]             "AYX/RAMX,CAT,AYX/@1,BYX/KMX,ALU/A+B,D_ALU
D_ACC               "UK/NOP"
D_D+LR              "RAMX/D,AMX/RAYX,BYX/LR,ALU/A+B,D_ALU"
D_D[].SC            "ALU_D[@1]RC[@2],[illegible]"
D_D.AND.R[]         "RBMX/D,BMX/RBYX,SPO.R/LOAD,LAB,SPO.RAB/@1,AYX/LA,ALU/AND,D_ALU"
D_D.LEFT.Q          "DX/LEFT,[illegible]"
D_D.OR.R[]          "RBYX/D,BMX/RBMX,SPO.R/LOAD,LAB,SPO.RAB/@1,AMX/LA,ALU/OR,D_ALU
D_D.EXT[]+K[]       "RAYX/D,AYX/RAMX,CAT,[illegible]/@1,KMX/@2,BMX/KYX,ALU/A+B,D_ALU"
D_D.XOR.RC[]        "RAYX/D,AYX/RAMX,SPO.R/LOAD,LC,SPO.RC/@1,BMX/LC,ALU/XOR,D_ALU"
D_LA+D              "AMX/LA,RBYX/D,BMX/RBYX,ALU/A+B,D_ALU"
D_LA.OR.D           "AMX/LA,RBYX/D,BYX/RBMX,ALU/OR,D_ALU"
D_MASK              "BYX/MASK,ALU/B,D_ALU"
D_MASK+1            "AYX/RAMX,CAT,BYX/MASK,ALU/A+B+1,D_ALU"
D_NOT.RC[]          "AYX/RBMX,[illegible],SPO.R/LOAD,LC,SPO.RC/@1,BYX/LC,ALU/BANDNOT,D_ALU"
D_Q.DP.D            "RAMX/Q,AMX/RAYX,ALU/[illegible],SHF/ALU,DX/SHF,BMX/RBYX,RBMX/D"
D_ND                "DX/NOP
D_R[].LEFT          "SPO.R/LOAD,LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/LEFT,DX/SHF"
D_R[].OR.D          "SPO.R/LOAD,LAB,SPO.RAB/@1,AMX/LA,RBYX/D,BMX/RBMX,ALU/OR,D_ALU"
```

```
D_R[].OPACT.MASK "S_U.R/LEA).[AR.SPF.RAR/#1,A/A/LA.BYA/YASK,ALU/BRNOT.D_ALU"
D_SC      "RMA/SC.ENA/BYA.ALU/B.S/*A.1,DA/SHF"
FA_U_STATE-R[]  "EBM-1RMA,RMA/#1,MSC/LOAD.STATE.EALL/A-B"
FBYA_R[]        "BYA/#1.LBYA/RMA"
FADDVR  "SUB/CALL.U/ZZZ"
ER_OOP  "SUB/CALL.U/FP.OOP"
ERROR1  "SUB/CAL_.U/FPRCP1"
ERROR2  "SUB/CAL_.U/FPRCP2"
LAB_R[FP4]      "SPF.AC/LDAD.LAB,SPD.ACN/PRR"
LAB_R[FP4+1]    "SPF.AC/LDAD.LAB,S'D.*CN/PRR-1"
LAB_R[SP1+1]    "S'D.AC/LDAD.LAB,SPD.AC4/SP1-1"
LAB_R[SSD]'1    "A'C.AC/ATOD.LAB,SPD.ACN11/SRC.DR.1"
LAB_R[SP1]      "SPF.AC/.TOD.LAB,STD.AC4/SP1.SP1"
LAB_R[SP1.SP1]  "SPF.AC/ TOD.LAB,STD.AC4/SP1.SP1"
YESSPARE1       "D_[.ZERO].CALL[YSUCUV]"
YESSAGE2        "D_[.1],CALL[YSUCUV]"
YESSAGE3        "D_[.2],CALL[YSUCUV]"
YESSAGE4        "D_[.3],CALL[YSUCUV]"
YESSAGE5        "D_R[.4],CALL[YSUCUV]"
YESSAGE6        "D_R[.5],CALL[YSUCUV]"
YESSAGE7        "D_R[.6],CALL[YSUCUV]"
YESSAGE8        "D_R[.7],CALL[YSUCUV]"
YESSAGE9        "D_R[.8],CALL[NSUCUV]"
YESSAGE10       "D_R[.9],CALL[NSUCUV]"
NEATST  "SUB/CALL.U/SOFTE"
NEATST[]        "NEATST.RC[SPI_A[#1]"
NOP     "DA/NOP"
Q_ADD           "DA/ALDEL"
Q_D.JQ          "BAMA/J,ANX,AAMQ,EPY/R2PX,REMA/Q_ALU/#1,Q_ALU"
Q_D.[PC]        "BAMA/J,ANX,PAM,SPD.R/LOAD,LC,SPJ.NC/PC,BMA/LC,ALU/#1,Q_ALL"
Q_[SP]          "RMA/SP1.CIN,DDX/ADX,ALU/B,Q_ALL"
Q_MASK=1        "AMA/R*VX.VT,EMY/YASK,ALL/ALS-1,Q_ALU"
Q_DA.mask       "AMA/R*VX.VT,EMY/YASK,ALL/ALS-1,Q_ALU"
Q_Q.AND.LC      "RAMA/Q,AMA/ BAYA BYA/LL,ALU/AND,Q_ALL"
Q_Q.AND.YASK    "RAMA/Q,AMA/ BAYA.YDX-YASK,ALU/AND,Q_ALL"
Q_Q.AND.R[]     "AMA/Q,BAY*BAYA.SPT.R/LOAD LAB,SPD.RAR/#1,AMX/LA,ALU/AND,Q_ALL"
Q_Q.ANDNOT.BU[] "AMA/Q,AYA/BAMA,SPT.R/LOAD,LC,SPU.RC/#1,SMX/LC,ALU/ANDNOT,Q_ALU"
Q_Q.LEFT2       "QX/.EFT2"
Q_Q.XOR.D       "RAMA/Q,AYA/BAMA,DAYA/J,DMX/PDYA,ALU/XOR,Q_ALU"
Q_Q.XOR.P[]     "DA/SHF,AYA/RAMA,RAMA/Q,DMX/LB,ALU/XOR,LAR_R[#1]"
RC[].D_R[]      "AMA/RAYA,DAYA.DMX.RC,CMD/WYA,ALL/A-S.RC[R1] ALU"
```

```
RC[]_D+Q           "RAMX/D,AMX/RAVX,REV*/Q,XVX/REMX,ALU/A+B,RC[#1]_ALU"
RC[]_D.AND.LA      "REMX/D,RMX/REVX,AVX/LA,ALU/AND,RC[#1]_ALU"
RC[]_D.ORNOT.MASK  "ALU_D.ORNOT.MASK,RC[#1]_ALU"
RC[]_D.XOR.LAB     "ALEG_D,BLEG_LB,ALU/XOR,RC[#1]_ALU"
RC[]_D.XOR.LC      "ALEG_D,BLEG_LC,ALU/XOR,RC[#1]_ALU"
RC[]_D+MASK+1      "RAVX/D,AVX/RAMX,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_LA+D          "AMX/LA,RBVX/D,BVX/REMX,ALU/A+B,RC[#1]_ALU"
RC[]_LAB.XOR.LC    "ALEG_LA,BLEG_LC,ALU/XOR,RC[#1]_ALU"
RC[]_MASK          "ALU_MASK,RC[#1]_ALU"
RC[]_NOT.D         "AVX/RAMX,EXT.DT/LONG,ALU/NOTA,RC[#1]_ALU"
RC[]_NOT.D         "ALU_NOT.D,RC[#1]_ALU"
RC[]_NOT.MASK      "BMX/MASK,AVX/RAMX,EXT.DT/LONG,ALU/ORNOT,RC[#1]_ALU"
RC[]_Q-LC          "RAMX/Q,AMX/RAVX,BMX/LC,ALU/A-B,RC[#1]_ALU"
RC[]_Q.AND.LA      "RAVX/Q,BMX/REMX,AMX/LA,ALU/AND,RC[#1]_ALU"
RC[]_Q.OR.LC       "RAVX/Q,AVX/RAMX,BMX/LC,BMX/RVX,ALU/OR,RC[#1]_ALU"
RC[]_Q+MASK        "RAVX/Q,AVX/RAMX,BMX/MASK,ALU/A+B,RC[#1]_ALU"
RC[]_Q+MASK+1      "RAVX/Q,AVX/RAMX,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/#1"
RC[]_Q.XOR.LAB     "ALEG_Q,BLEG_LB,ALU/XOR,RC[#1]_ALU"
RC[]_Q.XOR.LC      "ALEG_Q,BLEG_LC,ALU/XOR,RC[#1]_ALU"
RC[]_SC            "ALU_SC,RC[#1]_ALU"
RC[]_SHF           "SPO.R/WRITE.RC,SPO.RC/#1"
R[]_ALU.RIGHT2     "SHF/RIGHT2,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_D.AND.Q        "RAVX/D,AVX/RAMX,BMX/RBVX,ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_D.AND.RC[]     "RAMX/D,AVX/RAMX,SPO.R/WRITE.RC,SPO.RC/#2,BVX/LC,ALU/AND,R[#1]_ALU"
R[]_D.ANDNOT.MASK  "RAVX/D,AMX/RAMX,BMX/MASK,ALU/ANDNOT,R[#1]_ALU"
R[]_D.ANDNOT.Q     "RAMX/D,AMX/RAVX,RBMX/Q,BVX/REMX,ALU/ANDNOT,R[#1]_ALU"
R[]_D.LEFT         "RAMX/D,AMX/RAVX,ALU/A,SHF/LEFT,SPO.RAB/#1,SPO.R/WRITE.RAB"
R[]_D.OR.R[]       "RAMX/D,AMX/RAVX,RBX/#2,BMX/RMX,ALU/OR,R[#1]_ALU"
R[]_D.ORNOT.MASK   "RAVX/D,AVX/RAMX,BMX/MASK,ALU/ORNOT,R[#1]_ALU"
R[]_D.XOR.LAB      "ALEG_D,BLEG_LB,ALU/XOR,R[#1]_ALU"
R[]_D.XOR.LC       "ALEG_D,BLEG_LC,ALU/XOR,R[#1]_ALU"
R[]_LA.OR.R[]      "R[#2]_ALU,AMX/LA,BVX/RVX,BMX/#2,ALU/OR"
R[]_LAB.XOR.LC     "ALEG_LA,BLEG_LC,ALU/XOR,R[#1]_ALU"
R[]_NOT.LA         "AMX/LA,ALU/NOTA,SPO.R/WRITE.RAB,SPO.RAB/#1"
R[]_Q.XOR.LAB      "ALEG_Q,BLEG_LB,ALU/XOR,R[#1]_ALU"
R[]_Q.XOR.LC       "ALEG_Q,BLEG_LC,ALU/XOR,R[#1]_ALU"
R[]_SHF            "SPO.R/WRITE.RAB,SPO.RAB/#1"
RETURN             "SUB/RET,J/0"
SC_SC+1            "AVX/1,REMX/RVX,EALU/A+B,SMX/EALU,SCK/LOAD"
SETMEM[]           "R[VA]_LA,#1,CALL,J/SETMEM"
```

```
        RETRCF;         "XOR R[X ],CALL.U/RETRCF"
        RETRADS[]       R[0A]_R[9I],CALL.U/RETRADS"
        RETIADS[]       R[0A]_R[9I],CALL.U/RETIADS"
        SUBTest "CALL,,/SUBTST"
        TRAP.FPA        ACP/LOW.VC.,ACN/7"
        VA_D.DXT[]      ALU_D.BXT"9I",VA_ALU"
        VA_D.DXT[]+R[]  ALU_D.BXT"9I+4,9I",VA_ALU"
        VA_R[]+R[]      "CPD.R/LOAD.LAB.SPD.RA-/5",BMA/99,BMR/BNR.ADX/LA.U.U/A-B.VA_ALU"

; BRANCH DEFINITIONS
        ALL.??          "BFN/ALL"       ; NOTE ://XXB NESSECARY
        NIL.??          "BFK/ALL"
        01-0?           "BER/01-0"
```

DIAGNOSTIC SUPERVISOR COMMANDS

The diagnostic supervisor provides two major services. First, it allows the operator to control and run the macro-diagnostic programs through a command line interpreter. Second, it provides a program interface of common services required by all macro-diagnostic programs.

The diagnostic supervisor commands are grouped in three sets:

Program/test sequence control
Program flag control
Debug and utility features

Commands, switches and literal arguments may be abbreviated to the minimum number of characters necessary to retain their unique identity. For example, the LOAD command can be specified by a single 'L', whereas the START command requires a minimum of 'STA'.

In the symbolic command descriptions which follow, certain special characters are employed which require some explanation. Angle brackets, '<' and '>', are used to enclose symbolic arguments which are satisfied by a numeric expression or character string. Optional arguments are enclosed by square brackets, '[' and ']'. An octal function is indicated with '%'. Literal arguments such as ALL, OFF and FLAGS are capitalized.

PROGRAM/TEST SEQUENCE CONTROL COMMANDS

These commands enable the operator to select programs and portions of programs and to control the sequence of test execution. Note that the command line argument <file-name> refers to the five letter mnemonic code which identifies each diagnostic program.

Load Command

LOAD <File-Spec> [/PHYSICAL:<address>],

This command causes the specified file to be loaded into main memory. It can be used only when the Diagnostic Supervisor is run in the user mode (ODT link.) The starting address will be observed, if specified, but is normally omitted. If the address switch is used, it should be given in hexadecimal format.

NOTE

When diagnostic programs are run in the stand alone mode, both the diagnostic program and the diagnostic supervisor must be loaded with commands to the console program

Start Command

START [/SEQ:<section base>]-
[/TEST:<[!]-xx>[:<last>][/SUBTEST:<num>]-
[/PASS:<xxx:1>]

The "START" command causes the diagnostic supervisor to begin execution of the program in memory. As execution begins, the diagnostic supervisor enters into a dialogue with the operator to determine program specific parameters, such as, which unit is to be tested.

If the START command is given without switches, the program will run in the default section. The supervisor calls only those tests which have been selected by the program developer to run in the default section. Default section tests do not require operator intervention.

The /SECTION switch is program specific and not available for use with all programs. When a section is selected, only the tests which it contains will be executed.

The TEST switch is used in two distinctly different ways. If the "first" and "last" arguments are specified, the supervisor sequentially passes control to tests "first" through "last", inclusively. If the "first" argument is combined with the SUBTEST switch, program execution begins at the beginning of the "first" test and continues at the end of the subtest "num". If the SUBTEST switch is used in conjunction with the PASS switch, the operator is provided with a loop on subtest capability.

If the TEST switch is not specified, all tests within the default section of the program are executed. If only the "first" argument is specified with the TEST switch, the "last" argument is assumed by default to be the highest numbered test within the program.

Restart Command

```
RSTART [/SEC:<section-name>]-
       [/TEST:<first>[:<last>]/SUBTEST:<num>]]-
       [/PASS:<count>]
```

The RESTART command is similar to the START command. However, when using RESTART, the operator does not enter into the parameter retrieval dialogue with the program to select the device to be tested. This information must have been set up with a previous execution of the program in the START or RUN command line. The RESTART switches are identical to the START switches.

Run Command

```
RUN <file-spec> [/SEED: <section-name>]
    [/TEST:<first> [:<last>
    [/SUBTEST:<num>]] [/PASS:<count>]
```

The RUN command is available only when diagnostic programs are run under VAX. RUN is equivalent to a LOAD and START command sequence. The RUN command switches are identical to those in the START command.

Control C

The Control C returns control from a diagnostic program to the diagnostic supervisor. The supervisor then enters a command wait state. The operator may then issue any valid command.

Continue Command

```
CONT <CR>
```

This command causes program execution to resume at the point at which the program was suspended. This command is used to proceed from a breakpoint, error halt, or Control C situation.

Summary Command

```
SUMMARY
```

This command causes the execution of the programs summary report code section which prints statistical reports. Note that this command is generally used only after running a pass of a diagnostic program. However, the summary command can be used at any time, and could be useful, for example, when the Disk Reliability Program is run in the nonconversation mode.

Abort Command

ABORT

This command executes the program's cleanup code and returns control to the supervisor which enters a command wait state. At this point, the operator may issue any command except RESTART or CONTINUE.

EXECUTION CONTROL FUNCTIONS

The execution control functions allow the operator to alter the characteristics of the diagnostic programs and the diagnostic supervisor. These functions are implemented by flags which reside in the diagnostic supervisor and by event flags which are located within the programs. The flags are used to control the printing of error messages, ringing the bell, halting and looping of the program, etc.

Set Flags Command

SET [FLAGS] <arg-list>

This command results in the setting of the execution control flags specified by "arg-list". No other flags are affected. "arg-list" is a string of flag mnemonics from the following table, separated by commas.

HALT	Path on error detection. When the program detects a failure, if this flag is set, the supervisor enters a command wait state after all error messages associated with the failure have been output. The operator may then continue, restart or abort the program. This flag takes precedence over the LOOP flag.

LOOP — Loop on error. This flag, when set, causes the program to enter a predetermined error loop on a test or subtest which detects a failure. Looping will continue until the operator returns to the supervisor by using the Control C command. The operator may then continue, clear the flag and continue, restart, or abort the program.

BELL — Bell on error. This flag, when set, will cause the supervisor to output a "bell" to the operator whenever the program detects a failure.

IE1 — Inhibit error messages at level 1. When set, this flag suppresses all error messages, except those which are forced by the program or supervisor.

IE2 — Inhibit error messages at level 2. When set, this flag suppresses basic and extended information concerning the failure. Only the header (first three lines) information message is output for each failure.

IE3 — Inhibit error messages at level 3. When set, this flag suppresses extended information concerning the failure. The header and basic information messages are output for each failure.

IES — Inhibit summary report. When set, this flag suppresses additional report messages.

QUICK	Quick verify. This flag, when set, indicates to the program that the operator desires a quick verify mode of operation. The interpretation of this flag is program dependent.
SPOOL	Post error messages on line printer. This flag, when set, causes the supervisor to direct all program messages to the line printer. In the VMS environment, the messages are not actually printed but entered in a file on disk.
TRACE	Report the execution of each test. When set, this flag causes the supervisor to report the execution of each individual test within the program as the supervisor dispatches control to that test.
LOCK	Lock in physical memory. When set, this flag disables the program relocation function. Self-relocating programs are then locked into their current physical memory space.
OPERATOR	Operator present. When set, this flag indicates to the supervisor that operator interaction is possible. When cleared, supervisor takes appropriate actions to insure that the test session continues without an operator.
PROMPT	Displays long dialogue. When set, this flag indicates to the supervisor that the operator wants to see the limits and defaults for all questions printed by the programs.

ALL All flags in this list.

Clear Flags Command

CLEAR [FLAGS] <arg-list>

This command results in the clearing of the flags specified by "arg-list". No other flags are affected. Arg-list is a string of flag mnemonics separated by commas. See the SET command for supported arguments.

Set Flags Default Command

SET FLAGS DEFAULT

This command returns all flags to their initial default status. The default flag settings are OPERATOR and PROMPT.

Show Flags Command

SHOW FLAGS

This command displays all the execution control flags and their current status. The flags are displayed as two mnemonic lists; one list for those flags which are set, the other for those which are clear.

Set Event Flags Command

SET EVENT [FLAGS] <arg-list>[SET]

This command results in the setting of the event flags specified by "arg-list". No other event flags are affected. "Arg-list" is a string of flag numbers in the range of 1-23, separated by

commas. "ALL" may be specified instead of "arg-list". Note that event flags are program specific and that not all programs employ them.

Clear Event Flags Command

CLEAR EVENT [FLAGS] <arg-list>!ALL

This command results in the clearing of the event flags specified by "arg-list". No other event flags are affected. "Arg-list" is a string of flag numbers in the range of 1-23, separated by commas. An optional "ALL" may be specified instead of "arg-list".

Show Event Flags Command

SHOW EVENT [FLAGS]

This command causes the diagnostic supervisor to display a list of the event flags that are currently set.

DEBUG AND UTILITY FEATURES

This third set of commands enables the operator to alter diagnostic program code.

Set Base Command

SET BASE <virtual-address>

The command loads the address specified into a software register. This number is then used as a base to which the address specified in the SET BREAKPOINT, CLEAR BREAKPOINT, EXAMINE, and DEPOSIT commands is added. The SET BASE command is useful when referencing code in the diagnostic program listings. The base should be set to

the base address (see the program link map) of the program section referenced. Then the program counter (PC) numbers provided in the listings can be used directly in referencing locations in the program sections.

Note: Virtual addresses = Physical address (normally) when memory management is turned off.

Set Breakpoint Command

SET BREAKPOINT <address>

This command causes the diagnostic supervisor to assume control when the program accesses the location specified by the breakpoint address. A maximum of 15 breakpoints may be set within a diagnostic program.

Clear Breakpoint Command

CLEAR BREAKPOINT <address>|ALL

This command clears previously set breakpoints.

Show Breakpoints Command

SHOW BREAKPOINTS

This command displays all currently defined breakpoints.

Set Default Command

SET DEFAULT <arg-list>

This command results in the setting of default qualifiers for the EXAMINE and DEPOSIT commands. The "arg-list" argument consists of

a size default qualifier and/or a radix default qualifier, separated by a comma if both are present. If only one default qualifier is specified, the other one is not affected. Size default qualifier options include BYTE, WORD or LONG. Radix default qualifier options include HEXADECIMAL, DECIMAL and OCTAL. Default defaults are WORD and LONG.

Examine Command

EXAMINE [/<Qualifiers>] [<address>] [SKIP: <number>]
This command causes the diagnostic supervisor to display the contents of memory in the format specified by the qualifiers. The qualifiers are as follows:

/B	address points to a byte
/W	address points to a word
/L	address points to a longword
/X	display contents in hexadecimal radix
/D	display contents in decimal radix
/O	display contents in octal radix
/A	display contents as ASCII bytes

The "address" argument, when specified, is accepted in hexadecimal format, unless some other radix has been set with the DEFAULT command. Optionally, "address" may be specified as decimal or octal by immediately preceding it with "%D" or "%O", respectively. "Address" may also be one of the following: R0-R11, AP, FP, SP, PC, PSL.

Deposit Command

DEPOSIT [/<Qualifiers>] <address> <data>

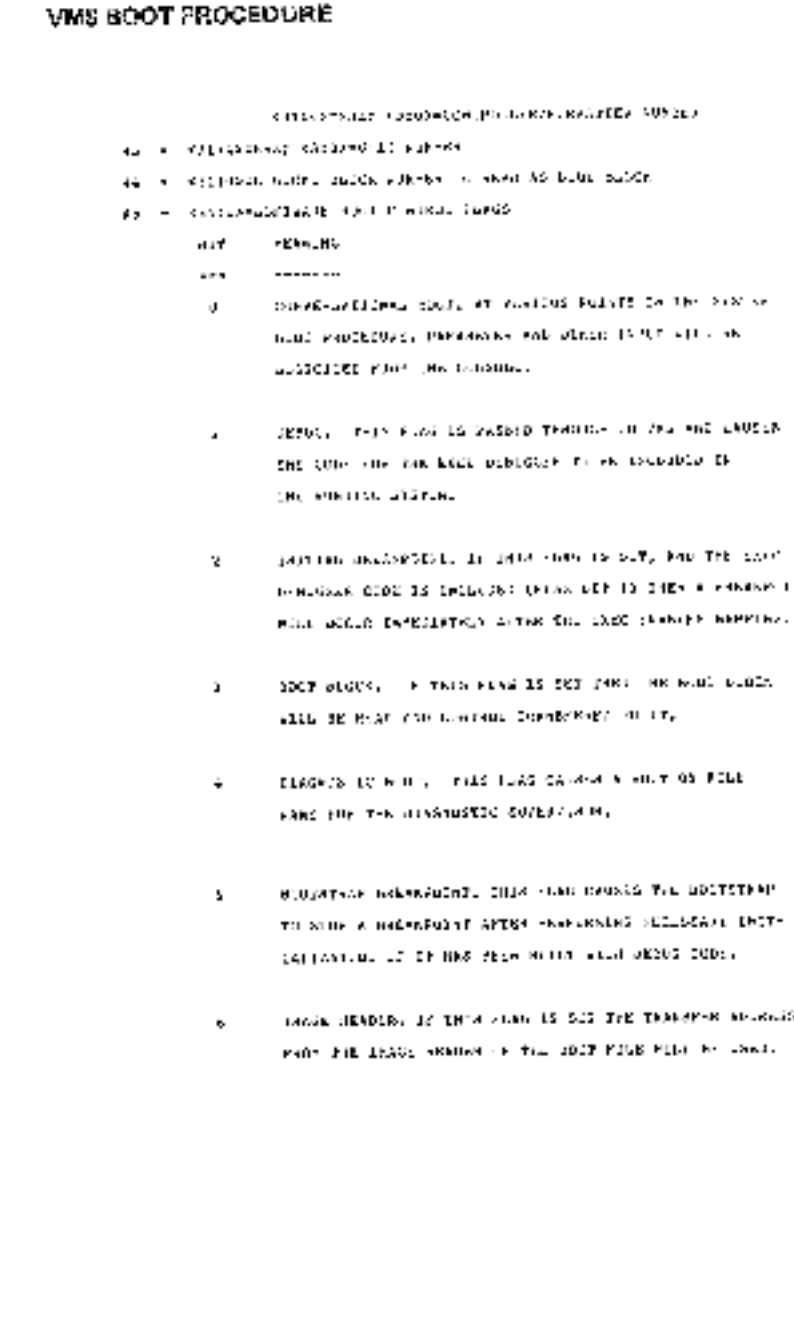
This command causes the diagnostic supervisor to accept data to be placed in memory at the specified address in the format described by the qualifiers. The qualifiers are as follows:

/B	address points to byte
/W	address points to word
/L	address points to longword
/X	accept data in hexadecimal radix
/D	accept data in decimal radix
/O	accept data in octal
/A	accept data in ASCII bytes

The "address" argument is accepted in hexadecimal format by default, unless some other radix has been set with the SET DEFAULT command. Optionally, "address" may be specified as decimal or octal by immediately preceding it with "%D" or "%O", respectively.

SECTION 7
SYSTEM OPERATION

```
$!
$! THIS FILE DESCRIBES THE INPUT PARAMETERS TO THE SYSTEM PROGRAM
$! VMB.EXE.  NORMALLY THE VAX/VMS BOOT WILL LOOKUP THE FILE [SYS0.SYSEXE]
$! OF THE SPECIFIED DEVICE.  HOWEVER IF ANY ERRORS ARE ENCOUNTERED DURING
$! IO THE
$!
$! THE VMB.EXE IS LOADED INTO MEMORY AT LOCATION 200.  THE FIRST
$! AVAILABLE VIRTUAL MEMORY IS ALSO USED BY THE SYSTEM PARAMETER
$! LOADR.  THE NUMBER OF THE PAGE OF THE VMB.EXE IS PASSED IN R5.
$! THE BLOCK NUMBER, WHERE IT ALSO SEARCH AS A PRIMARY STACK IF ERROR
$!
$ SYS$INPUT:
$!
$! NOTE:  FILENAME, DIRECTORY AND TYPE MUST
$!
$!       X => DISK NAME          (MAXIMUM OF 4 CHARACTERS)
$!       Y => DIRECTORY NAME     (MAXIMUM OF 39 CHARACTERS)
$!
$! P1  =  FILENAME,FILETYPE (OR ALTERNATE FILE NAME)
$!        P2 MUST CONTAIN BOOTING FLAGS.  (SEE VAX/VMS SYSTEM MANAGER'S
$!        GUIDE)
$!
$!               IF NEEDED       MAXIMUM # OF CHARACTERS
$!               ---------       -----------------------
$!                   A           DEVICE NAME
$!                   B           PASSWORD NUMBER (OPTIONAL)
$!                   C           PASSWORD NUMBER (OPTIONAL)
$!
$! P2  =  FLAG VALUE
$!
$!               (SEE VAX/VMS SYSTEM MANAGER'S GUIDE FOR DETAILS)
$!
$!               PASSWORD
$!               ACCOUNT NUMBER = [0,4]
$!
$! EOD SYS$
```



7-3

OPERATOR CONTROL WILL TRANSFER TO THE FIRST KEY OF THE
BOOT FILE.

7 MINIMUM TEST INHIBIT. THIS FLAG INHIBITS THE TESTING
OF PREVIOUSLY GOOD/ERRORFREE.

A PROMPT FOR, CAUSES THE OPERATOR TO SELECT THE NAME
OF THE BOOT FILE.

B HALT BEFORE PROCEED, CAUSES A HALT INSTRUCTION
TO BE EXECUTED PRIOR TO THE TRANSFER TO THE BOOTFILE.
THIS OPTION IS USEFUL FOR DEBUGGING PURPOSES.

D0 NO DEC DIAGNOSTIC. A FLAG PASSED TO/FROM TO THE BOOTFILE
THAT PREVENTS EXECUTION OF DEC SPECIFIC FUNCTIONALITY
RECEIVED FROM THE TOOL OR AVAILABLE HERE.

R#1 - HALT #1

R#1 - HALT #0,

AP - HALT @ R6

SP - SOURCE(?)(20) OF FIRST WORDING BOOT RETURN ADDRESS

USABLE AS BOTH SCRATCH POINTER AND POINTER TO GOOD MEMORY.

USE OF FILEX TO TRANSFER DIAGNOSTIC FILES

In order to load and run diagnostic programs under the VMS operating system, the operator may have to transfer diagnostic files from the floppy disk to the VMS pack on a system device. Proceed as follows. Any terminal on the system may be used.

1. Insert diskette ZZ-E8ZE8 into the floppy disk drive. The 5 position key switch on the control panel can be in either the LOCAL or the LOCAL DIAGNOSTIC position.

2. Type in the following commands under VMS.

```
^Y                              ; Control Y to return control to
                                ; VAX/VMS.

$ ALL DXA1:                     ; Allocate the floppy disk.
  _DXA1 ALLOCATED

$ MOUNT/FOR DX1:                ; Mount the floppy disk.

$ MCR FLX                       ; Invoke the FILEX program.  FILEX
                                ; displays the prompt symbol, FLX>.

FLX>/DX/ASCII:<CODE>.BIN/RT/IM
                                ; Transfer the file from the floppy
                                ; disk to the system device, where
                                ; <CODE> is the mnemonic of the
                                ; program.  Transfer each file
                                ; needed in this manner.
```

```
FLX>/CO/BL:512./RB-DX1:ESSAA.K8K/BC/IM
                                ; transfer the diagnostic
                                ; supervisor.
FLX> ^Y                         ; Return control to VAX/VMS.

$ DISM DX1:                     ; Dismount the floppy disk.
$ DEALL DX1:                    ; Deallocate the floppy disk.
$                               ; VAX/VMS                    prompt
```

3. Sample console terminal output:

```
$ MCR FLX
FLX> /CO/BL:512./RB-DX1:ESSAA.K8K/BC/IM
FLX> /RB-DX1:ESMMA.K8K/BC/IM
FLX> ^Y
$
```

TERMINAL FUNCTION KEYS

RETURN		(Carriage return.) Transmits the current line to the system for processing. (On some terminals, the RETURN key is labeled CR.)
		Before a terminal session, initiates login sequence.
Control characters		Define functions to be performed when the CTRL key and another key are pressed simultaneously. All CTRL/x key sequences are echoed on the terminal as ^x.
	CTRL/C*	During command entry, cancels command processing.
		Before a terminal session, initiates login sequence.
	CTRL/I	Duplicates the function of the TAB key.
	CTRL/K	Advances the current line to the next vertical tab stop.
	CTRL/L	Form feed.
	CTRL/O	Alternately suppresses and continues display of output to the terminal.
	CTRL/Q	Restarts terminal output that was suspended by CTRL/S.
	CTRL/R	Retypes the current input line and leaves the cursor positioned at the end of the line.
	CTRL/S	Suspends terminal output until CTRL/Q is pressed.
	CTRL/U	Cancels the current line and discards it.
	CTRL/Y	Interrupts command or program execution and returns control to the command interpreter.
	CTRL/Z	Signals end-of file for data entered from the terminal.
DELETE		Deletes the last character entered at the terminal and backspaces over it. (On some terminals, the DELETE key is labeled RUBOUT.)
ESCAPE		Has special uses in particular commands or programs, but generally performs the same function as RETURN. (On some terminals, the ESCAPE key is labeled ALTMODE.)
TAB		Moves the printing element or cursor on the terminal to the next tab stop on the terminal. Most terminals have tab stops at every 8 character positions on a line.

COMMANDS FOR TERMINAL COMMUNICATION AND CONTROL

:=	(String assignment.) Defines a local symbol name as a synonym for a DCL command.
:==	(String assignment.) Defines a global symbol name as a synonym for a DCL command.
HELP	Displays information about a command or a command parameter or qualifier on the terminal.
LOGOUT	Terminates communication between a user and the system.
MCR	Passes a command line to the MCR-like MCR environment, or places the terminal in MCR command mode.
REQUEST	Displays a message at an operator's terminal.
SET PASSWORD	Changes the password associated with your user name for subsequent logins.
SET TERMINAL	Defines the characteristics of the terminal for the duration of a terminal session.
SHOW DAYTIME	Displays the current date and time of day on the terminal.
SHOW SYMBOL	Displays current local or global symbols and the strings or values assigned to them.
SHOW TERMINAL	Displays the current characteristics of the terminal.

COMMANDS FOR FILE MANIPULATION

APPEND	Adds the contents of one or more files to the end of another file.
COPY	Copies one or more files into another file.
CREATE	Creates a file from data entered at the terminal or in the input stream.
CREATE/DIRECTORY	Defines a new directory or subdirectory for cataloging files.
DELETE	Removes a directory entry for a file or files and makes any data in the file(s) inaccessible.
DELETE/ENTRY	Deletes an entry from the print or batch job queue.
DIFFERENCES	Compares the contents of files and produces a report indicating the differences between the two.
DIRECTORY	Displays information about a file or a group of files.
DUMP	Displays the contents of a file in binary format.
EDIT	Begins an interactive editing session to create or modify a file.
PRINT	Queues a copy of a file for printing on the system printer.
PURGE	Deletes all but the most recent version or versions of a specified file or files.
RENAME	Changes the name of a file or a group of files.
SET DEFAULT	Changes the default directory and/or disk device used to identify files.
SET QUEUE/ENTRY	Changes the attributes or status of an entry in the printer queue.
SET PROTECTION	Changes the protection applied to a file or a group of files, restricting or allowing access to the file by different categories of user.

COMMANDS FOR FILE MANIPULATION

SHOW DEFAULT	Displays the current default directory and disk device.
SHOW PROTECTION	Displays the default protection applied to new files created.
SHOW QUEUE	Displays the names of files queued to the printer and not yet printed, or the names of jobs submitted for batch execution but not yet processed.
SORT	Creates a sorted copy of a file, with records arranged in a particular collating sequence.
SORT/RSX11	Invokes the SORT-11 program to create a sorted copy of a file.
STOP/ABORT	Halts printing of a file that is currently being printed.
TYPE	Displays the contents of a file or files at the terminal.
UNLOCK	Allows access to a file that was not properly closed.

COMMANDS FOR DEVICE HANDLING

ALLOCATE	Reserves a device for use by a single user and, optionally, assigns a logical name to the device.
ASSIGN	Defines a file specification or a device name to be associated with a logical name for subsequent use in commands or programs.
DEALLOCATE	Relinquishes use of a previously allocated device, thus making the device available to other users.
DEASSIGN	Cancels a logical name assignment made with the ALLOCATE, ASSIGN, DEFINE, or MOUNT commands.
DISMOUNT	Releases the connection between a user and a disk or tape volume that is currently mounted on a device.
INITIALIZE	Deletes all existing data, if any, on a mass storage volume and readies the volume for new data.
MOUNT	Makes a disk or tape volume available for the reading and writing of files, and optionally assigns a logical name to the device on which the volume is mounted.
SHOW DEVICES	Displays devices currently in use by the process, or system devices available for use.
SHOW LOGICAL	Displays current logical name assignments for a particular logical name table.
SHOW TRANSLATION	Searches all three logical name tables for a logical name and displays the equivalence name of the first match found.

ASSIGN	Defines a file specification to be associated with a specific logical name used in a program.
BASIC	Invokes the PDP-11 BASIC-PLUS 2/VAX compiler to compile BASIC language source statements.
CANCEL	Halts periodic execution of an image scheduled for execution in a process.
COBOL/RENAME	Invokes the PDP-11 COBOL-74/VAX compiler to compile a set of COBOL language source statements.
CONTINUE	Resumes execution of an interrupted command, program, or command procedure.
DEBUG	Invokes the VAX/VMS Symbolic Debugger to begin interactive debugging.
DEFINE	Equates character strings with logical names. These names can be assigned and translated from within user programs.
DEPOSIT	Replaces the contents of a location in virtual memory with new data or instruction.
EDIT	Invokes an editor to create or modify a source program or data file.
EXAMINE	Displays the contents of a location in virtual memory.
FORTRAN	Invokes the VAX-11 FORTRAN IV-PLUS compiler to compile a set of FORTRAN language source statements.
LIBRARY	Creates or modifies a macro library or a library of object modules.
LINK	Binds one or more object modules into an executable or shareable program image.
LINK/BOOT	Invokes the MCA-11M Task Builder to link one or more object modules into an executable task image.
MACRO	Invokes the VAX-11 MACRO assembler to assemble a set of MACRO language source statements.

COMMANDS FOR PROGRAM DEVELOPMENT AND CONTROL

MACRO/RSX	Invokes the MACRO-11 assembler to assemble a set of assembler language source statements.
PATCH	Updates an image file.
RUN (Image)	Places an executable image in execution in the current process.
RUN (Process)	Creates a separate process to execute a specified image.
SHOW LOGICAL	Displays on the terminal the current assignments of logical names and equivalence names made by the ASSIGN, ALLOCATE, DEFINE or MOUNT commands.
SHOW PROCESS	Displays information about the current process, including subprocesses, privileges, quotas, and accounting information.
SHOW STATUS	Displays information about the image currently executing in the process.
SHOW SYSTEM	Displays the current status of all processes in the system.
SHOW TRANSLATION	Displays the result of logical name translation of a specific logical name.
STOP (Image)	Halts execution of a command procedure, program, or a subprocess.

@file-spec	(Execute Procedure). Executes a command procedure or places data in a command file in the input stream.
=	(Arithmetic assignment.) Equates a local symbol name to an arithmetic expression or constant.
==	(Arithmetic assignment.) Equates a global symbol name to an arithmetic expression or constant.
:=	(String assignment.) Equates a local symbol name to any character string.
:==	(String assignment.) Equates a global symbol name to any character string.
DECK	Marks the beginning of records to be read as the input data stream for a command. (Required only when data contains dollar signs in the first position of any record.)
DELETE/ENTRY	Deletes a job from the batch job queue.
EOD	Marks the end of an input data stream begun with the DECK command.
EOJ	Signs the end of a batch job submitted through a system card reader.
EXIT	Terminates a command procedure.
GOTO	Transfers control to another statement in a command procedure.
IF ... THEN	Tests symbolic value or command or program status value and performs stated action based on the result of the test.
INQUIRE	Requests interactive assignment of a variable value for a symbolic name.
JOB	Marks the beginning of a batch job submitted through a system card reader.
ON ... THEN	Defines the action to be taken when a command or program incurs errors of particular severity levels.

PASSWORD	Provides a password associated with a job entered through the system card reader.
SET NOCK	Suppresses command interpreter error checking following command execution.
SET NOVERIFY	Suppresses display of command lines executed in command procedures subsequently executed.
SET QUEUE/[NAME]	Changes the characteristics of a queued batch job.
SET VERIFY	Causes all command lines in command procedures subsequently executed to be displayed at the terminal or printed in the batch job log file.
STOP	Terminates command procedure processing at any level and returns control to the command interpreter.
SUBMIT	Queues a command procedure to the batch job queue.

USER ENVIRONMENT TEST PACKAGE (UETP)

1. Log out from the field service account.

   $ LOGOUT

   The system responds:

   VAX/VMS LOGOUT at 12:43:10 17-JUL-1978

2. Log in to the SYSTEST account.

   <CR>

   USERNAME: SYSTEST

   PASSWORD: UETP

3. Prepare the devices for testing. For each disk drive to be tested (not the system load device), perform the following steps:

   Physically mount a scratch disk. Set the drive to VON. Issue the following commands.

   $ INIT/DATA_CHECK DMA0: TEST1
   $ MOUNT/SYSTEM DMA0: test1
   $ CREATE/DIRECTORY DMA0: SYSTEST
   $ CREATE/DIRECTORY DMA0: 1, 7

   Note

   When repeating this set of commands for each disk drive on the system, be sure to specify the device name (eg. DMA0:)

For each magnetic tape drive perform the following steps:

Physically mount a write enabled scratch tape at least 600 feet long. Press the ONLINE switch.

For each hard copy terminal and line printer, check the paper supply (2 pages for one pass of the UETP).

Press the ONLINE switch. Check the baud rates and terminal characteristics (these should still be set according to specifications given in section 6.16, above).

4. Run the entire UETP by entering the UETP command procedure and responding to the three prompts, as shown below.

```
$ @SYS$UPDATE:UETP TEST*DATA

VAX/VMS UETP STARTED dd-MMM-yy hh:mm

ENTER NUMBER OF LOAD TEST CYCLES (D): 0
ENTER NUMBER OF CONCURRENT UETP test JOBS (D): 1
ENTER SCRATCH MAGTAPE (S,D, ETC:) OR ALLOW device-name

0:11>
```

NOTE

The following table specifies responses to the load test number question, according to memory size.

Guidelines for Selecting

Number of Load Test Users

Size of Memory	Number of Load Test Users
MS Based Systems	
256K	10
384K	15
512K	20
640K	25
768K	30
896K	35
1 megabyte	40
MS Based Systems	
256K	6
384K	9
512K	12
640K	15
768K	18
896K	21
1 megabyte	25

c. Check the operator terminal output for errors. Indication of errors in this output (short file) can be followed up with an examination of the output file specified in the UETP command for ICBECBACK. In addition, DEVELOG.LOG is a large log file containing a concatenation of individual log files from the following tests:

```
        the I/O device tests
        the native mode utility tests
        the system load test
        the compatibility mode tests
```

3. If it becomes necessary to interrupt or terminate the UETP run, use the following commands, as appropriate.

```
First, type
        ^Y              ; Control-Y interrupts the current UETP test
                        ; and temporarily returns control to the
                        ; command language interpreter.

then
        STOP            ; Terminate execution of the UETP.

or
        CONTINUE        ; Continue the test from the point of
                        ;                               interruption.
```

PRINTING THE ERROR LOG FILE

This procedure shows the operator how to create an error log report and how to obtain a copy of it. This procedure does not explain the mechanics of the RUN SYS$SYSTEM:SYS command.

Procedure:

1. Set the default disk to the system disk and to the default directory [SYSERR] by typing the following commands:

```
$ SET DEFAULT SYS$SYSTEM
$ SET DEFAULT SYS$SYSDEVICE:[SYSERR]
```

2. Examine the [SYSERR] directory to see what versions of the ERRLOG.SYS file are on disk by typing:

```
$ DIRECTORY ERRLOG.SYS
```

3. Rename the latest version of the ERRLOG.SYS file to ERRLOG.OLD by issuing the command:

```
$ RENAME ERRLOG.SYS ERRLOG.OLD/NEW_VERSION
```

4. Invoke the SYS utility by typing the command:

```
$ RUN SYS$SYSTEM:SYS
```

PRINTING THE ERROR LOG FILE

Prompts

```
Input File:     input-file-spec
Output File:    output-file-spec
Option:         report-option
Device Name:    device-name[:]
After:          date time
Before:         date time
```

Command Parameters

input-file-spec

Specifies an error log file created by the ERRFMT process. By default, SYS uses the highest version of the [SYSERR]ERRLOG.OLD which resides on the system disk.

When the user explicitly specifies an input file, SYS defaults any omitted fields to the defaults for DUP/UAI unit 1, that is, the highest version of SYS$DISK:[default-directory]ERR001.DAT.

output-file-spec

Specifies the file to contain the error log report. By default, SYS sends the output to SYS$OUTPUT, which is usually the user's terminal.

If the user explicitly specifies an output file, SYS defaults any omitted fields to the defaults for DUP/UAI unit 2, that is, the highest version of SYS$DISK:[default-directory]ERR002.DAT.

/report option

One of the following report options:

R Roll up
C Cryptic
B Brief
S Standard
U Unknown

By default, SYE uses the Roll up (R) option when none other is specified.

The five report options are explained in the Description section.

/device=name:

Instructs SYE to report only errors encountered by the specified device type or device unit. By default, SYE reports errors on all devices.

SYE prompts for the device name by typing

device name: [<all>] ?

by default, errors on all devices are reported (i.e., if only a carriage return is typed, all error types are inspected).

If a device name is specified, then device errors and mount/dismount messages whose device names match are selected for further inspection.

SYS will accept generic device names, allowing the operator to specify that errors be reported for all devices of a particular type (e.g., DB:), for devices attached to a particular controller (e.g., DSA:), or for a particular device (e.g., DNA1:).

When you specify a device name to SYS, you may also request that it report one or three special classes of errors:

CR	Hardware errors other than device errors, including machine checks, Corrected Read Data, Scsd Data Substitutes, SDI alerts, SDI faults and Asynchronous Write errors.
CO	Configuration changes, including mount and dismount messages.
SY	System information, including system startup, power recovery, crash/restart, system service and network messages, and system and user bugchecks. Although time stamp messages are included in the summary totals under system information, they are not included in this option.

Finally, you may also use the device name to deselect one device class or special class by prefixing the name with a minus sign( ).

For example:

```
     device name:      (all)     ? -DMA1:
```

means output all errors other than DMA1: errors. -SD would means all errors except system information entries to be reported.

Only one device or special class can be deselected at a time.

Date time

Instructs SYE to report on errors occurring after a specified date and time (AFTER: absolute time) and/or before a specified date and time (BEFORE: absolute time).

By default, SYE reports errors after 17 NOV 1858 and before 31-DEC-9999.

Be certain to enter the following file name in response to the input file prompt (Input file:).

```
SPRLOG.OLD
```

This is the file created in step 3 of this procedure.

5. Obtain a copy of the error log report by entering the following command:

```
$ PRINT filename
```

Note that this is more necessary if default output is used, because the file would have been listed on the terminal.

The filename is the name of the file entered in response to the output file prompt (output file:).

Examples:

```
$ SET DEFAULT SYS$SYSTEM
$ SET DEFAULT SYS$DISK:[SYSERR]          $ SHOW DEFAULT
$ DIRECTORY ERRLOG.SYS                   $ DUA2:[SYSERR]

DIRECTORY DUA2:[SYSERR]
18-JUL-78 15:11

ERRLOG.SYS;1    14.   18-JUL-78    13:48

TOTAL OF 14./18. BLOCKS IN 1. FILE
$ RESAVE ERRLOG.SYS ERRLOG.OLD/NEW_VERSION
$ RUN SYS$SYSTEM:SYS
SYS XX.X-X
 Input file:   (SYSERR)ERRLOG.OLD    ?<CR>
_Output file:  [SYS$OUTPUT]          ERRLOG.DAT
_Options:      [BRIEF-HF]            HR
_Device name:  [<all>]                ?<CR>
_aClass name:  [(1-NOV-1858)          ?<CR>
                                       (7-NOV-1858 00:00:00.00
_eClass date:  [31-DEC-9999]         ?<CR>
                                       31-DEC-9999 23:59:59.
```

PRINTING THE ERROR LOG FILE

```
Successful completion
$ PRINT ERRLOG.DAT
```

The SAT DEFAULT commands set the operator's default disk and directory to ERR2:[SYSTEM]; the response to the SHOW DEFAULT command verifies this. The DIRECTORY command lists, on the operator's terminal, all the ERRLOG.SYS files contained in the [SYSERR] directory; ERRLOG.SYS;1 file is the only file named. The RENAME command renames ERRLOG.SYS to ERRLOG.OLD. The /NEW_VERSION qualifier requests that ERRLOG.OLD be assigned a new version number if a file of this name already exists. The operator then invokes the SYE utility by typing RUN SYS$SYSTEM:SYE. SYE prompts for six parameters. The first prompt requests the name of the file to be manipulated by SYE; the response is <CR>, which follows the default of ERRLOG.OLD. The second prompt requests the file to contain the error log report; the response is ERRLOG.DAT. The third prompt requests the type of report that SYE generates; the response is the HOLE UP (R) report. The fourth prompt requests the devices on which SYE reports errors; the response, <CR> (which has SYE should report errors on all devices. The fifth and sixth prompts request the time range in which errors are recorded; the responses <CR> indicate that SYE should report all errors that occur between 17-NOV-1900 and 31-DEC-9999. SYE then creates the error log report and notifies the operator at completion. The operator can then issue the PRINT command to obtain a hard copy of this report.

PRINTING THE ERROR LOG FILE

SYR Report types

- Roll up -- A roll up report is a summary of errors. For each failing device or system component covered in the report, the report totals the number of hardware and software errors. The summary does not include any information about individual errors.

- Brief -- A brief report includes a descriptive, but brief, entry for each error covered in the report. Each entry describes, at least, the type of error, the device or component that caused the error, the error's sequence number, and when the error was logged.

- Cryptic -- A cryptic report applies to device and CP hardware errors only. For each error included in the report, the report shows the contents of associated registers every time an error occurred. The register contents are shown without explanation.

- Standard -- A standard report contains an entry for every error; each entry includes all the information captured about the error. In addition, each item of information has a corresponding English explanation.

  A standard report is 72 or more wide, and is therefore suitable for displaying at a terminal.

* Unknown -- An unknown error report documents unknown, invalid, and undefined errors. Such errors include errors on devices not recognized by SYE. In addition, an error can be classified as Unknown when the system information garnered to describe the error has been corrupted in some way. This report uses the standard format.

Additional SYE Rules

1. SYE internal errors that are not file open errors are reported via SORTAN error messages. If an error message occurs, rerun SYE to verify that the error was not covered by an operator error.

2. A Read data substitute (RDS) is logged twice per error, once as a memory error, and once as a machine check. The machine check information does not contain the memory controller register contents. RDSs are therefore logged as memory errors when the entries in the error log will be consecutive, with the time of error being the same.

3. On memory errors (i.e., CRD, SBI, ALERT, RDS), registers for all memory controllers on the system are listed. The operator must determine which controller is at fault.

4. On a system bugcheck or crash/restart, with an error message code of "Unexpected Unibus adapter interrupt", the general registers (R0-R5), dumped, contain the following information.

R0=UBA Configuration Register

R1=UBA Control Register

R2=UBA Status Register

R3=UBA Diagnostic Control Register

R4=UBA Failed Map Address Register

R5=UBA Failed Datum Address Register

5. TU78 Magtape errors may be reported as TU16 errors.

SECTION 8
CONVERSION TABLES
AND INTEGRATED CIRCUIT DIAGRAMS

HEX ADDER

+	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	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	+0
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	+0	+1
3	3	4	5	6	7	8	9	A	B	C	D	E	F	+0	+1	+2
4	4	5	6	7	8	9	A	B	C	D	E	F	+0	+1	+2	+3
5	5	6	7	8	9	A	B	C	D	E	F	+0	+1	+2	+3	+4
6	6	7	8	9	A	B	C	D	E	F	+0	+1	+2	+3	+4	+5
7	7	8	9	A	B	C	D	E	F	+0	+1	+2	+3	+4	+5	+6
8	8	9	A	B	C	D	E	F	+0	+1	+2	+3	+4	+5	+6	+7
9	9	A	B	C	D	E	F	+0	+1	+2	+3	+4	+5	+6	+7	+8
A	A	B	C	D	E	F	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
B	B	C	D	E	F	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A
C	C	D	E	F	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B
D	D	E	F	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C
E	E	F	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D
F	F	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E

HEX SUBTRACTER

	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	-1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
2	-2	-1	0	1	2	3	4	5	6	7	8	9	A	B	C	D
3	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	A	B	C
4	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	A	B
5	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	A
6	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
7	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
8	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
9	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
A	-A	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5
B	-B	-A	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4
C	-C	-B	-A	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3
D	-D	-C	-B	-A	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2
E	-E	-D	-C	-B	-A	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1
F	-F	-E	-D	-C	-B	-A	-9	-8	-7	-6	-5	-4	-3	-2	-1	0

	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	10	11	12	13	14	15
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
20	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
30	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
40	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
50	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
60	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
70	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
80	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
90	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D0	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

HEX/OCTAL CONVERSION

	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	10	11	12	13	14	15	16	17
10	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37
20	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57
30	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77
40	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117
50	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137
60	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157
70	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177
80	200	201	202	203	204	205	206	207	210	211	212	213	214	215	216	217
90	220	221	222	223	224	225	226	227	230	231	232	233	234	235	236	237
A0	240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257
B0	260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277
C0	300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317
D0	320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337
E0	340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357
F0	360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	377

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	10	11
10	12	13	14	15	16	17	20	21	22	23
20	24	25	26	27	30	31	32	33	34	35
30	36	37	40	41	42	43	44	45	46	47
40	50	51	52	53	54	55	56	57	60	61
50	62	63	64	65	66	67	70	71	72	73
60	74	75	76	77	100	101	102	103	104	105
70	106	107	110	111	112	113	114	115	116	117
80	120	121	122	123	124	125	126	127	130	131
90	132	133	134	135	136	137	140	141	142	143
100	144	145	146	147	150	151	152	153	154	155
110	156	157	160	161	162	163	164	165	166	167
120	170	171	172	173	174	175	176	177	200	201
130	202	203	204	205	206	207	210	211	212	213
140	214	215	216	217	220	221	222	223	224	225
150	226	227	230	231	232	233	234	235	236	237
160	240	241	242	243	244	245	246	247	250	251
170	252	253	254	255	256	257	260	261	262	263
180	264	265	266	267	270	271	272	273	274	275
190	276	277	300	301	302	303	304	305	306	307
200	310	311	312	313	314	315	316	317	320	321
210	322	323	324	325	326	327	330	331	332	333
220	334	335	336	337	340	341	342	343	344	345
230	346	347	350	351	352	353	354	355	356	357
240	360	361	362	363	364	365	366	367	370	371
250	372	373	374	375	376	377				

HEXADECIMAL/ASCII CONVERSION

HEX Code	ASCII Char	HEX Code	ASCII Char	HEX Code	ASCII Char	HEX Code	ASCII Char
00	NUL	20	SP	40	@	60	`
01	SOH	21	!	41	A	61	a
02	STX	22	"	42	B	62	b
03	ETX	23	#	43	C	63	c
04	EOT	24	$	44	D	64	d
05	ENQ	25	%	45	E	65	e
06	ACK	26	&	46	F	66	f
07	BEL	27	'	47	G	67	g
08	BS	28	(	48	H	68	h
09	HT	29	)	49	I	69	i
0A	LF	2A	*	4A	J	6A	j
0B	VT	2B	+	4B	K	6B	k
0C	FF	2C	,	4C	L	6C	l
0D	CR	2D	-	4D	M	6D	m
0E	SO	2E	.	4E	N	6E	n
0F	SI	2F	/	4F	O	6F	o
10	DLE	30	0	50	P	70	p
11	DC1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	73	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	SUB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D	}
1E	RS	3E	>	5E	^	7E	~
1F	US	3F	?	5F	_	7F	DEL

25S10 FOUR BIT SHIFTER CHIP WITH TRISTATE OUTPUT

LOGIC SYMBOL

LOGIC DIAGRAM

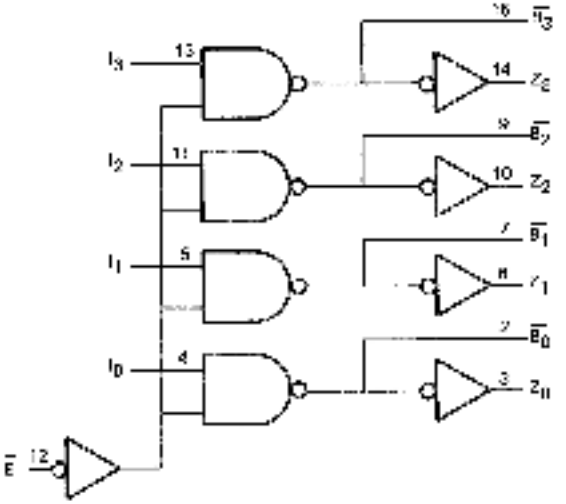
TRUTH TABLE

H	X	X	X	X	X	X	X	X	X	X	Z	Z	Z	Z
L	L	L	P_0	P_1	P_2	P_3	X	X	X	X	P_0	P_1	P_2	P_3
L	L	H	X	P_0	P_1	P_2	P_3	X	X	X	P_1	P_2	P_3	L
L	H	L	X	X	P_3	P_2	P_1	P_0	X	X	P_2	P_3	L	L
L	H	H	X	X	X	P_3	P_2	P_1	P_0	X	P_3	L	L	L

H = HIGH X = DON'T CARE

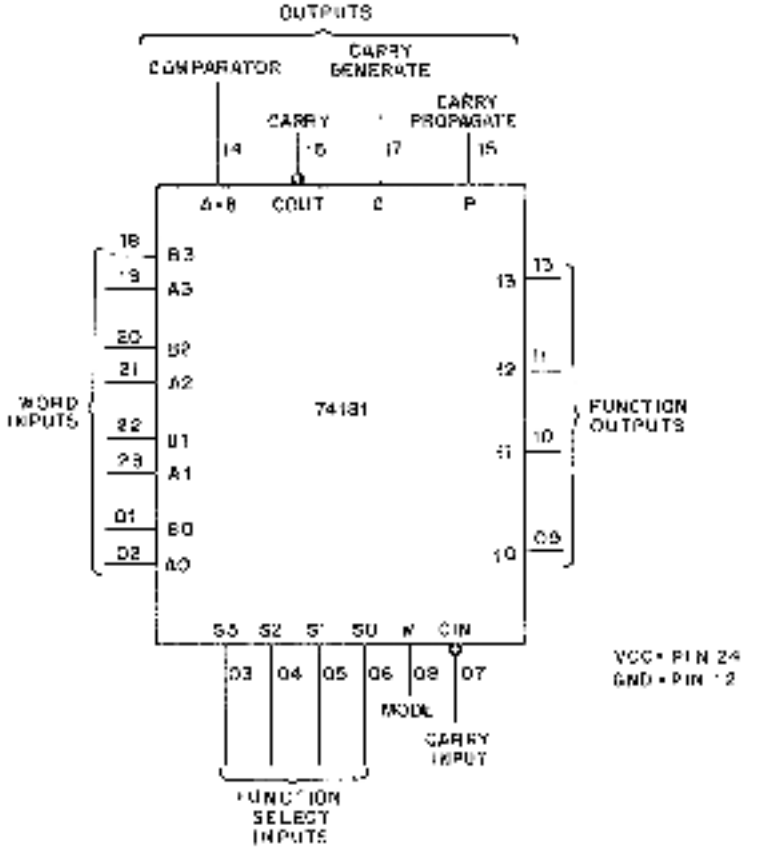
L = LOW Z = HIGH IMPEDANCE (OFF)

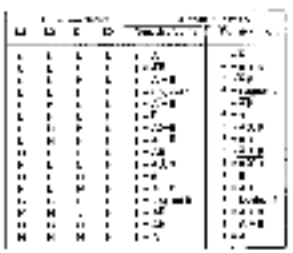
26S10 BUS TRANSCEIVER CHIP



OUTPUTS		INPUTS		H = HIGH L = LOW X = DON'T CARE Y = VOLTAGE OF BUS (ASSUMES CONTROLLED BY ANOTHER BUS TRANSCEIVER)
$\overline{E}$	D	$\overline{E}$	A	
L	L	H	L	
L	H	L	H	
H	X	Y	$\overline{Y}$	

74LS181 ALU CHIP

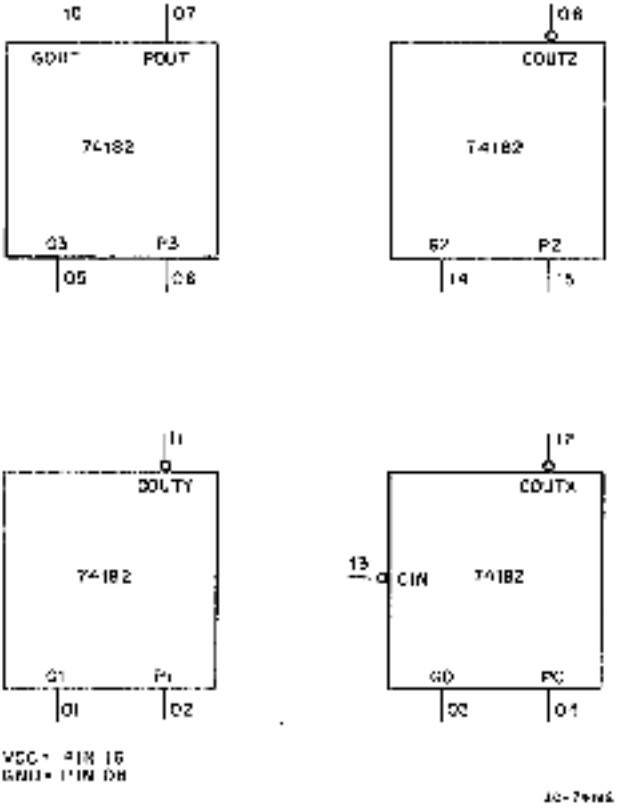




74182 LOOK AHEAD CARRY CHIP

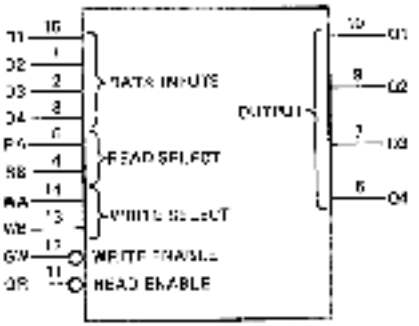
PIN DESIGNATIONS

Designation	Pin No.	Function
G0, G1, G2, G3	3, 1, 14, 5	ACTIVE-LOW CARRY GENERATE INPUTS
P0, P1, P2, P3	4, 2, 15, 6	ACTIVE-LOW CARRY PROPAGATE INPUTS
CIN	13	CARRY INPUT
COUTX, COUTY, COUTZ	12, 11, 9	CARRY OUTPUTS
GOUT	10	ACTIVE-LOW CARRY GENERATE OUTPUT
POUT	7	ACTIVE-LOW CARRY PROPAGATE OUTPUT
V_{CC}	16	SUPPLY VOLTAGE
GND	8	GROUND



VCC = PIN 16
GND = PIN 08

74LS670 4 X 4 REGISTER FILE CHIP



PIN DESIGNATIONS

WRITE FUNCTION TABLE
(SEE NOTES A, B, AND C)

WRITE INPUTS			WORD			
W_B	W_A	$\overline{G_W}$	0	1	2	3
L	L	L	Q_0	Q_1	Q_2	Q_3
L	H	L	D_0	Q_1	Q_2	Q_3
H	L	L	Q_0	D_1	Q_2	Q_3
H	H	L	D_0	D_1	D_2	D_3
X	X	H	Q_0	Q_1	Q_2	Q_3

READ FUNCTION TABLE
(SEE NOTES A AND D)

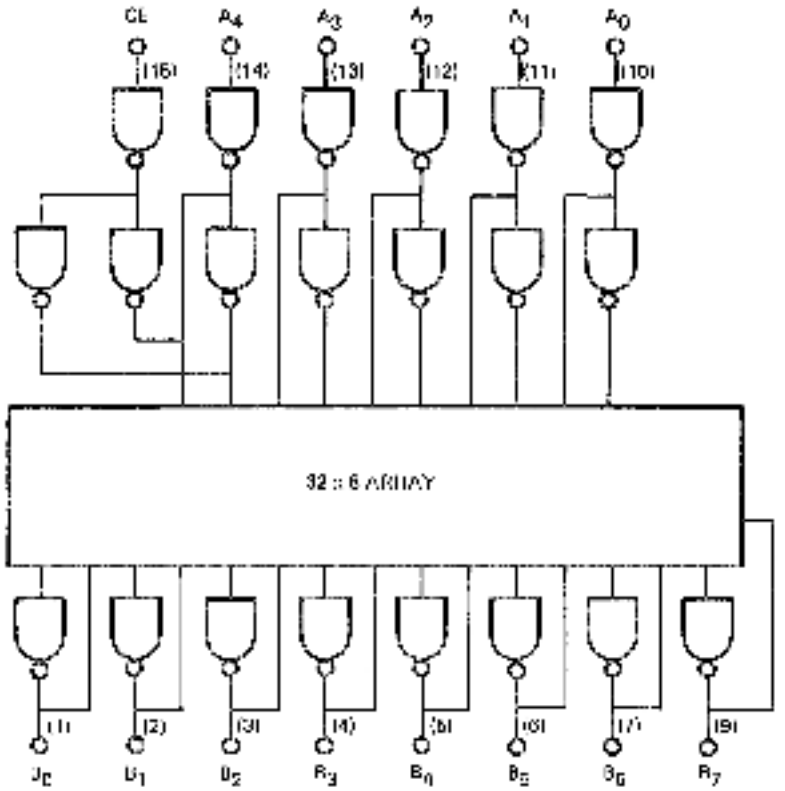
READ INPUTS			OUTPUTS			
R_B	R_A	$\overline{G_R}$	O1	O2	O3	O4
L	L	L	W0B1	W0B2	W0B3	W0B4
L	H	L	W1B1	W1B2	W1B3	W1B4
H	L	L	W2B1	W2B2	W2B3	W2B4
H	H	L	W3B1	W3B2	W3B3	W3B4
X	X	H	Z	Z	Z	Z

A. H = HIGH LEVEL, L = LOW LEVEL, X = IRRELEVANT, Z = HIGH IMPEDANCE (OFF)

B. Q_0 = D_0 = THE FOUR SELECTED INTERNAL FLIP-FLOPS OUTPUTS WILL ASSUME THE STATES APPLIED TO THE FOUR EXTERNAL DATA INPUTS.

C. Q_0 = THE LEVEL OF Q BEFORE THE INDICATED INPUT CONDITIONS WERE ESTABLISHED.

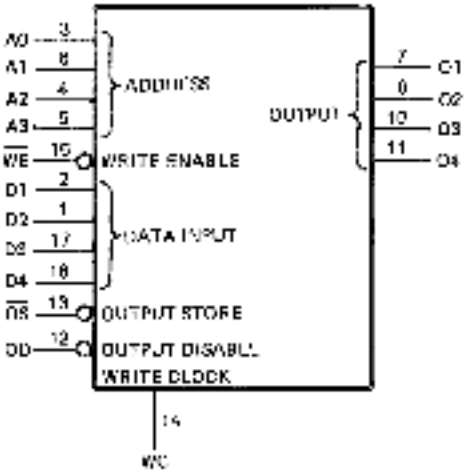
D. W0B1 = THE FIRST BIT OF WORD 0, ETC.



THE 82S22 USES OPEN COLLECTOR OUTPUTS.
THE 82S123 USES TRISTATE OUTPUTS.

V_{CC} = (15)
GND = (8)
(N) = DENOTES PIN NUMBERS

93S68 64 BIT EDGE TRIGGERED D TYPE REGISTER FILE CHIP WITH TRISTATE OUTPUTS

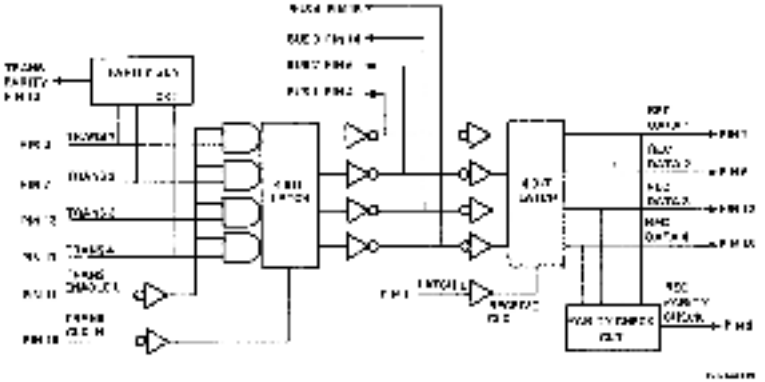


VCC 16
GND 8

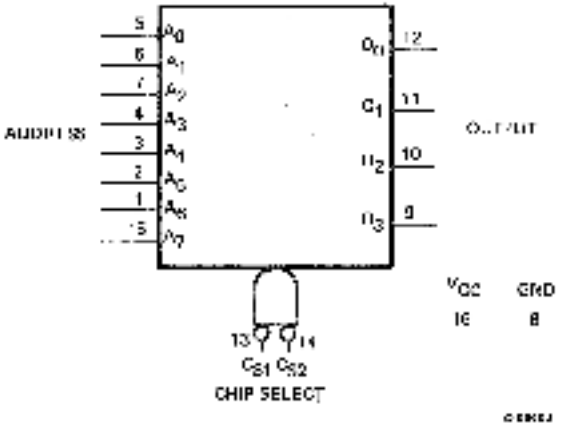
TRUTH TABLE

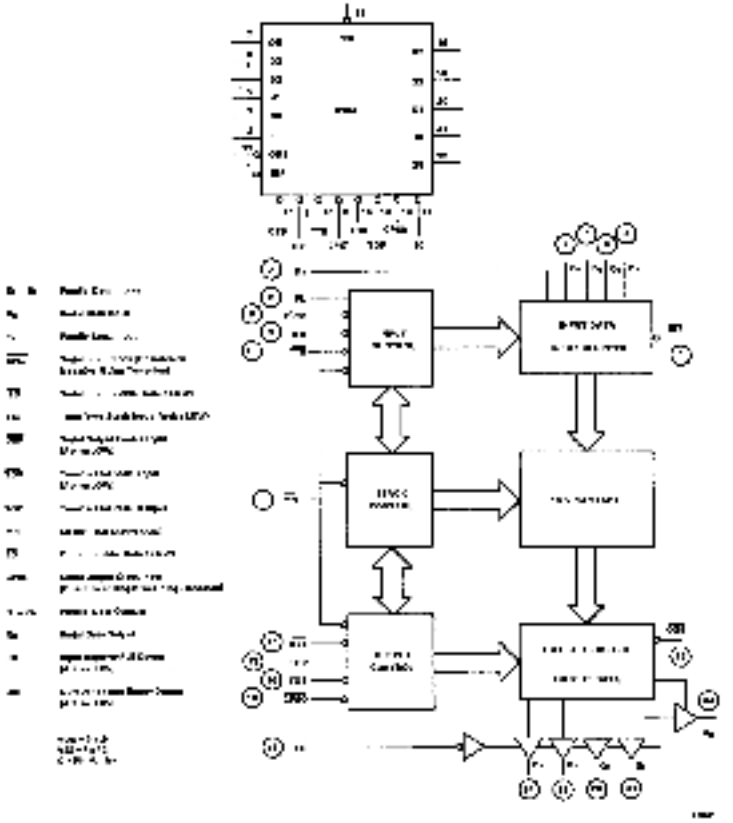
OD	$\overline{WE}$	CLK	$\overline{OS}$	MODE	OUTPUTS
L	X	X	L	OUTPUT STORE	DATA FROM LAST ADDRESSED LOCATION
X	L	⊔	X	WRITE DATA	DEPENDENT ON STATE OF OD AND $\overline{OS}$
L	X	X	H	READ DATA	DATA STORED IN ADDRESSED LOCATION
H	X	X	L	OUTPUT STORE	HI-Z
H	X	X	H	OUTPUT DISABLE	HI-Z

DEC 9646 4 BIT TRISTATE BACKPLANE INTERCONNECT TRANSCEIVER CHIP

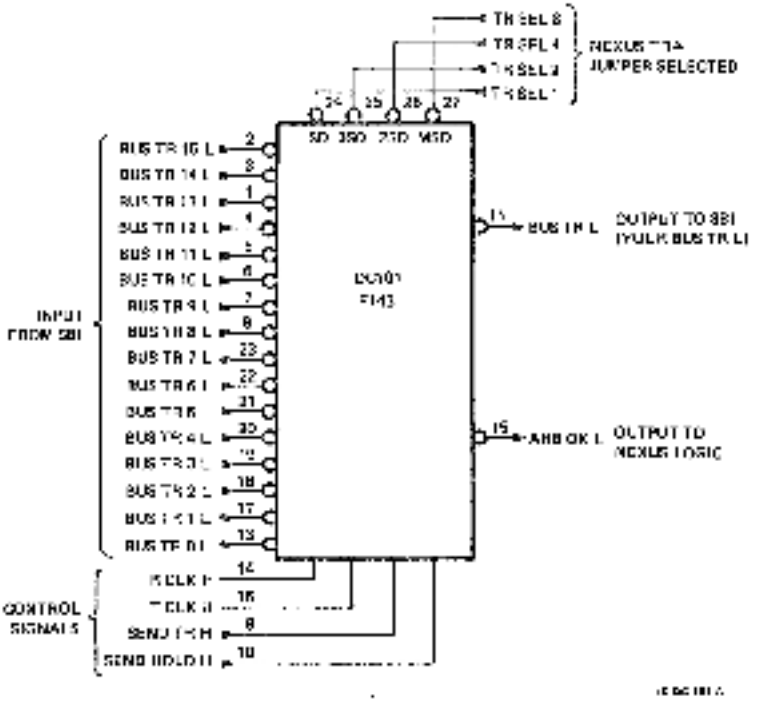


93406 1024 BIT ROM CHIP

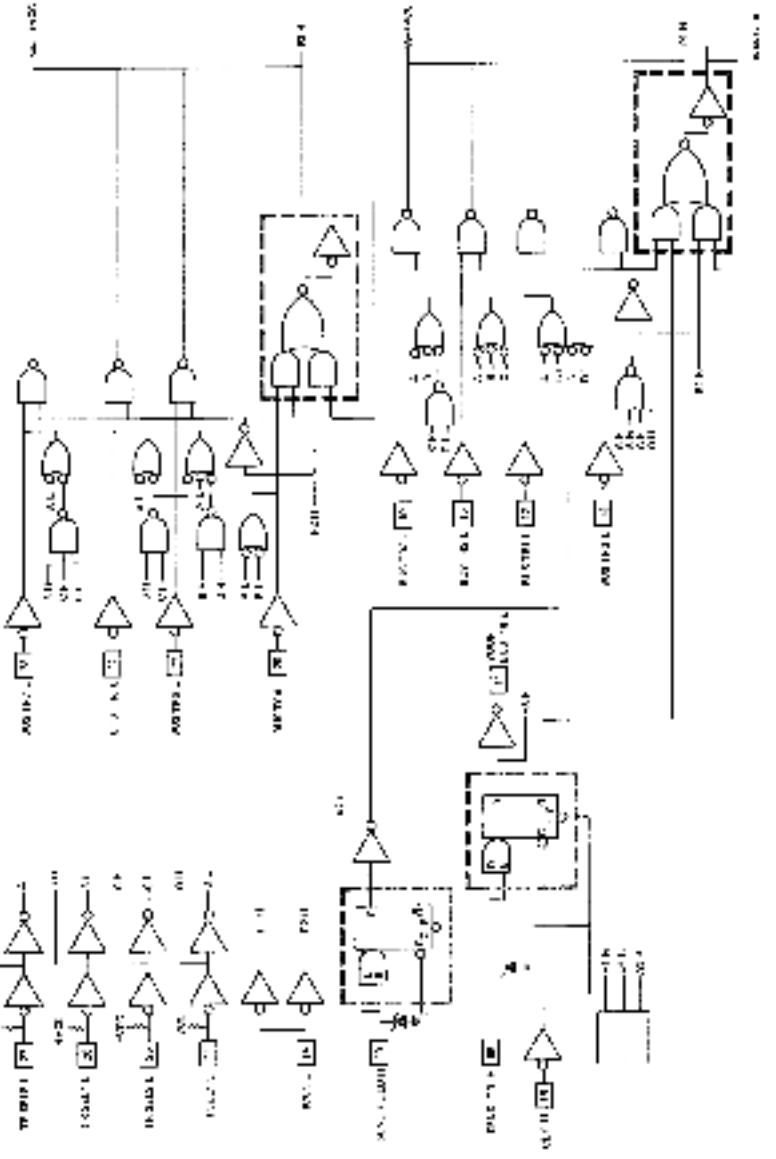




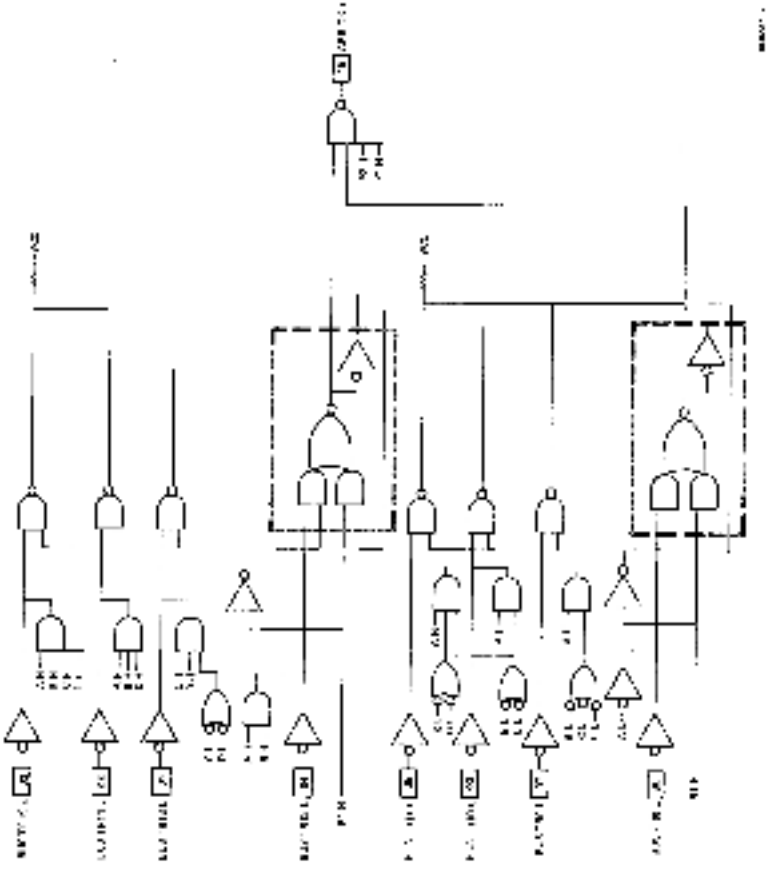
DC101 ARBITRATOR CHIP, PART 1

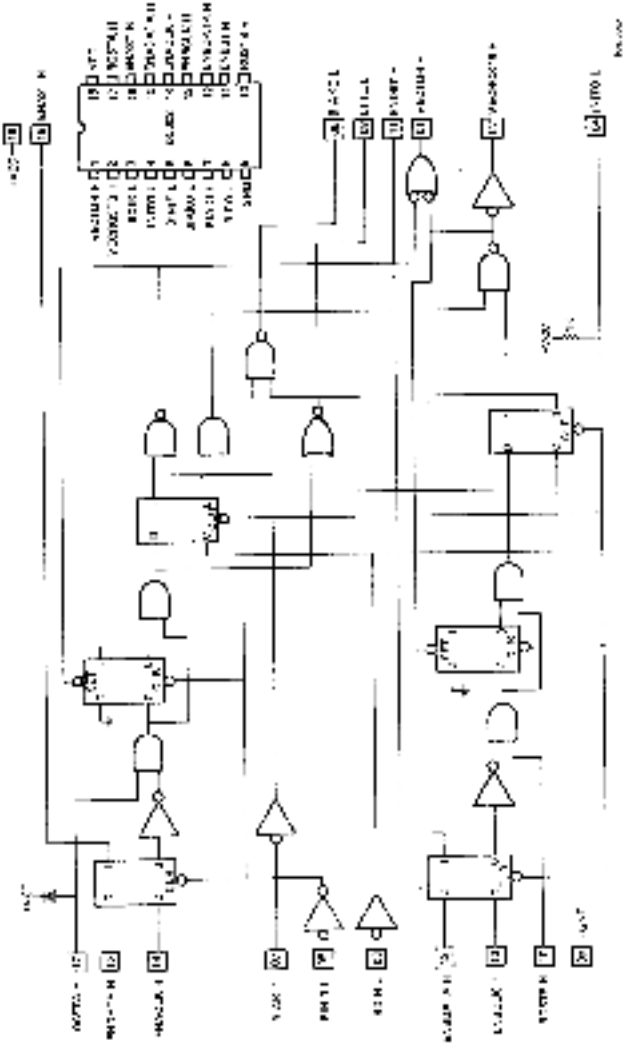


DG101 ARBITRATOR CHIP, PART 2

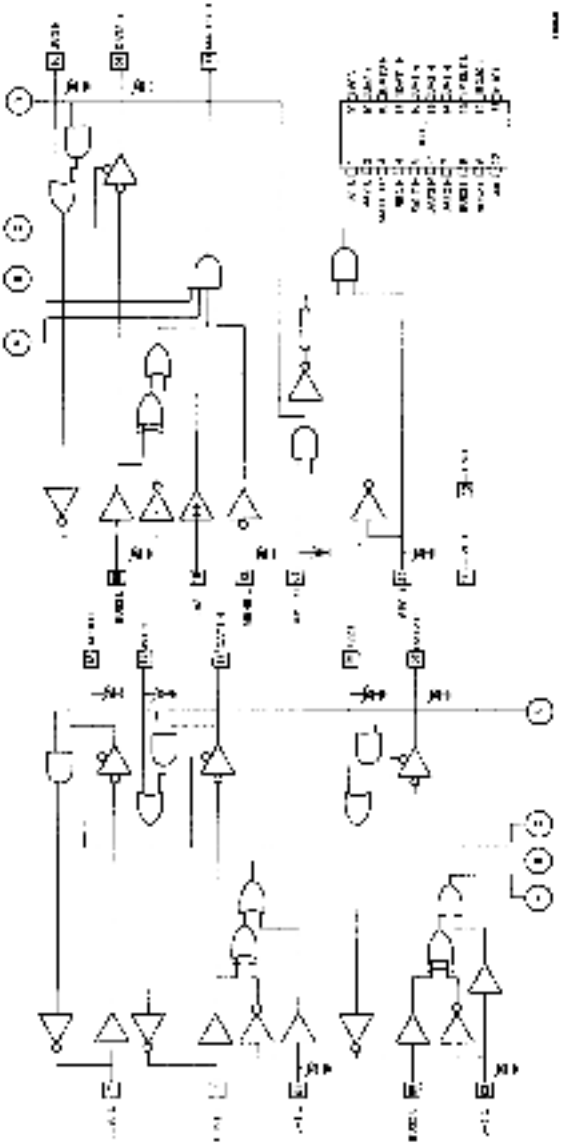


DC101 ARBITRATOR CHIP, PART 2





CX9004 PROGRAMMABLE CHIP



NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

VAX-11/780
SYSTEM MAINTENANCE GUIDE
EK-11780-PS-001

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to the manual? Is your judgment in complete, accurate, well organized, well written, etc.? Is it easy to use?

What features are most useful?

What faults or errors have you found in the manual?

Does this manual satisfy the need you think it was intended to satisfy?

Does it satisfy your needs? Why?

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532
Attention: Communications Services (NR2/N15)
Customer Services Section

Order No. EK-11780-PS-001

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
Maynard, Massachusetts 01754

digital