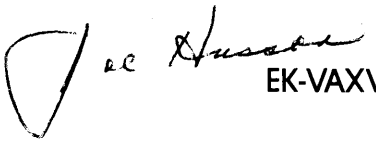


EK-VAXV2-HB-002

VAX Maintenance Handbook

VAX-11/780

1983 Edition

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, August 1982
Second Edition, March 1983

Printed in U.S.A.

CONTENTS

CHAPTER 3 MICROCODE

CHAPTER 4 CONSOLE

CHAPTER 5 INTERNAL REGISTERS

CHAPTER 6 SYSTEM BACKPLANE INTERCONNECT (SBI)

CHAPTER 7 SBI NEXUS

CHAPTER 1

INTRODUCTION

INTRODUCTION

The purpose of the VAX Maintenance Handbook is to provide a compact, quick-reference source of troubleshooting, maintenance, operating, and programming information that is frequently referenced by DIGITAL field service, manufacturing, training, and engineering personnel.

This second volume of the VAX Maintenance Handbook is devoted exclusively to information on the VAX-11/780 processor.

VAX-11/780 HARDWARE MANUALS

Title	Document Number
VAX-11/780 Power System Technical Description	EK-PS780-TD-001
VAX-11/780 System Installation Manual	EK-SI980-IN-001
DS780 Diagnostic System User's Guide	EK-DS780-UG-001
DS780 Diagnostic System Technical Description	EK-DS780-TD-002
FP780 Floating-Point Processor Technical Description	EK-FP780-TD-001
REP05/REP06 Subsystem Technical Documentation	EK-REP06-TD-001
VAX-11/780 Central Processor Technical Description	EK-KA780-TD-001
VAX-11/780 Memory System Technical Description	EK-MS780-TD-00
DW780 UNIBUS Adapter Technical Description	EK-DW780-TD-001
KC780 Console Interface Technical Description	EK-KC780-TD-001
VAX-11/780 Software Handbook	EB08126
VAX-11/780 Architecture Handbook	EB07466
VAX-11/780 Multiport Memory Subsystem	E7-MA780-TD-001
DR780 General Purpose Interface User's Guide	EK-DR780-UG-001
VAX-11/780 TB, Cache, SBI Control Technical Description	EK-MM780-TD
VAX-11/780 RH780 Technical Description Manual	EK-RH780-TD
VAX Diagnostic System User's Guide	EK-VX11D-UG-001

Hardcopy manuals can be ordered from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532

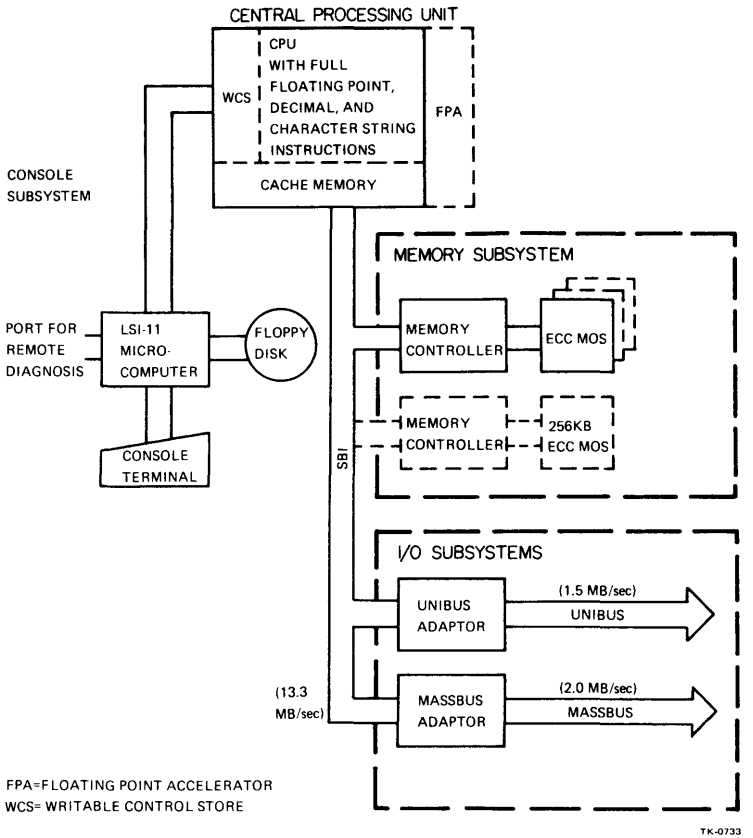
Attention: Printing and Circulation Services (NR2/M15)
Customer Services Section

For information concerning microfiche libraries, contact:

Digital Equipment Corporation
Micropublishing (E46)
12 Crosby Drive
Bedford, MA 01730

CHAPTER 2
HARDWARE DIAGRAMS

VAX-11/780 BLOCK DIAGRAM

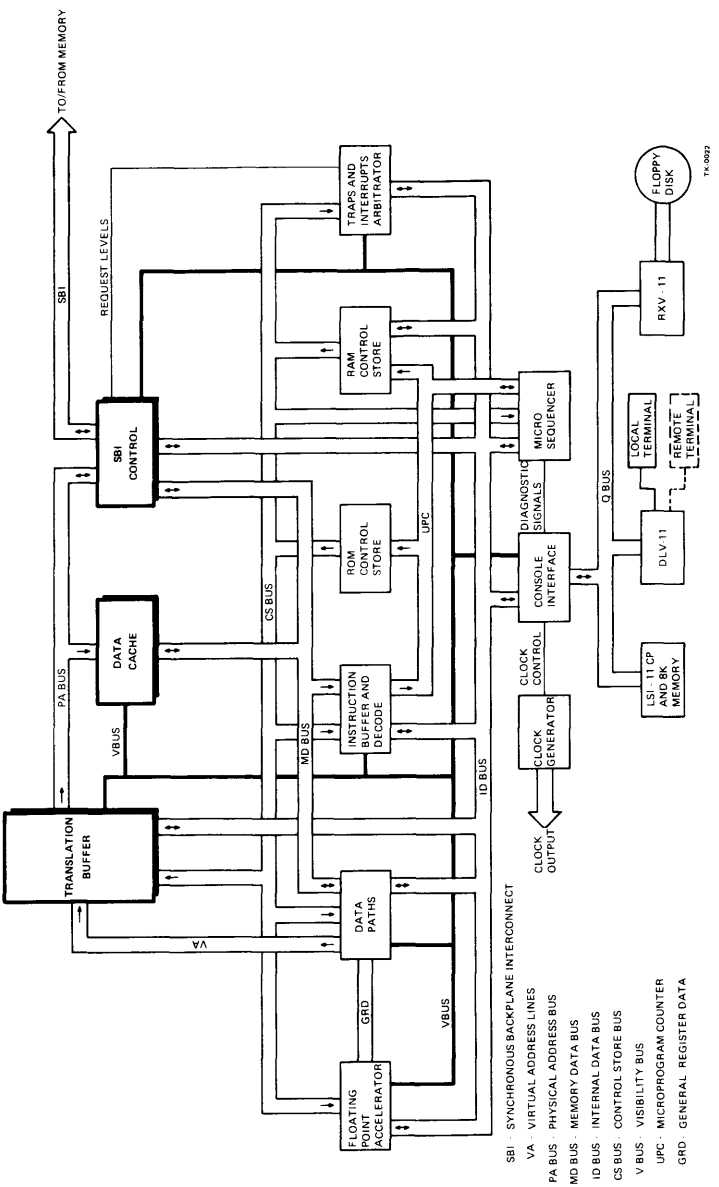


KA780 CPU MODULE UTILIZATION CHART

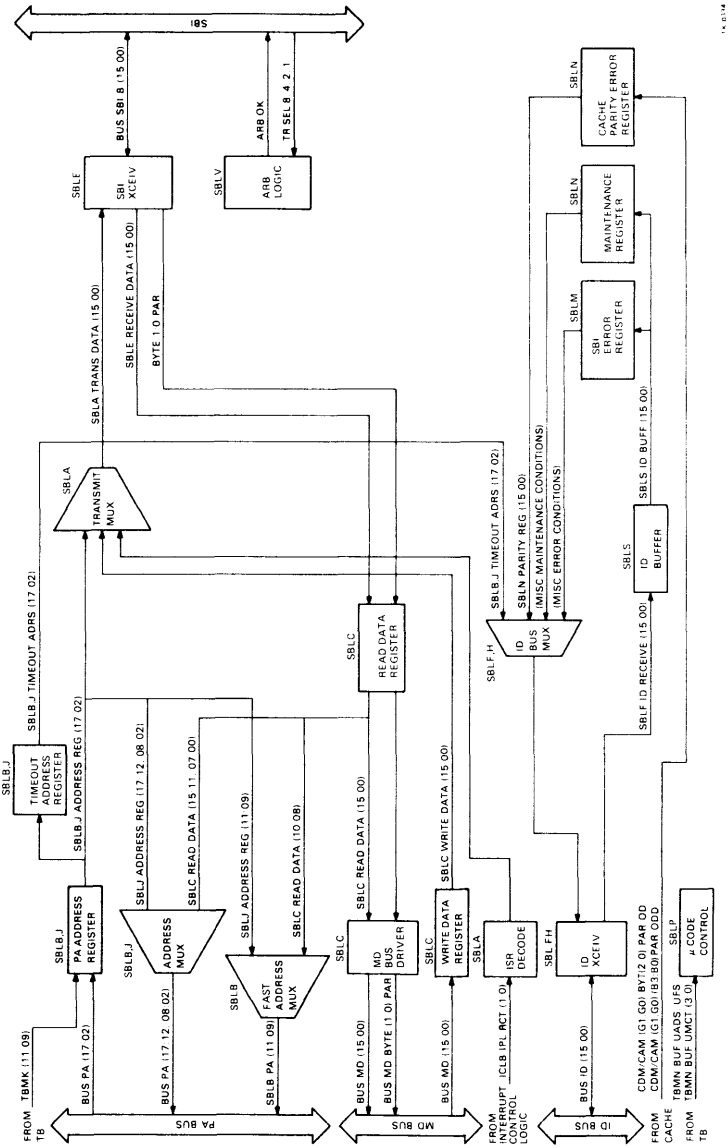
MODULE UTILIZATION KA780			
29	M8236	CIB	
28	M8289	FCT	•
27	M8288	FAD	•
26	M8287	FML	•
25	M8286	FMH	•
24	M8285	FNM	•
23	M8235	USC	
22	M8234	PCS	
21			
20	M8233 or M8238	WCS	
19			
18	M8233 or M8238	OCS	•
17			
16	M8232	CLK	
15	M8231	ICL	
14	M8230	CEH	
13	M8229	DAP	
12	M8228	DCP	
11	M8227	DDP	
10	M8226	DEP	
9	M8225	DBP	
8	M8224	IRC	
7	M8223	IDP	
6	M8222	TBM	
5	M8221	CDM	
4	M8220	CAM	
3	M8219	SBH	
2	M8218	SBL	
1	M8237	TRS	
• WHEN NOT INSTALLED USE BLANK MODULE 7014103			

TK-8346

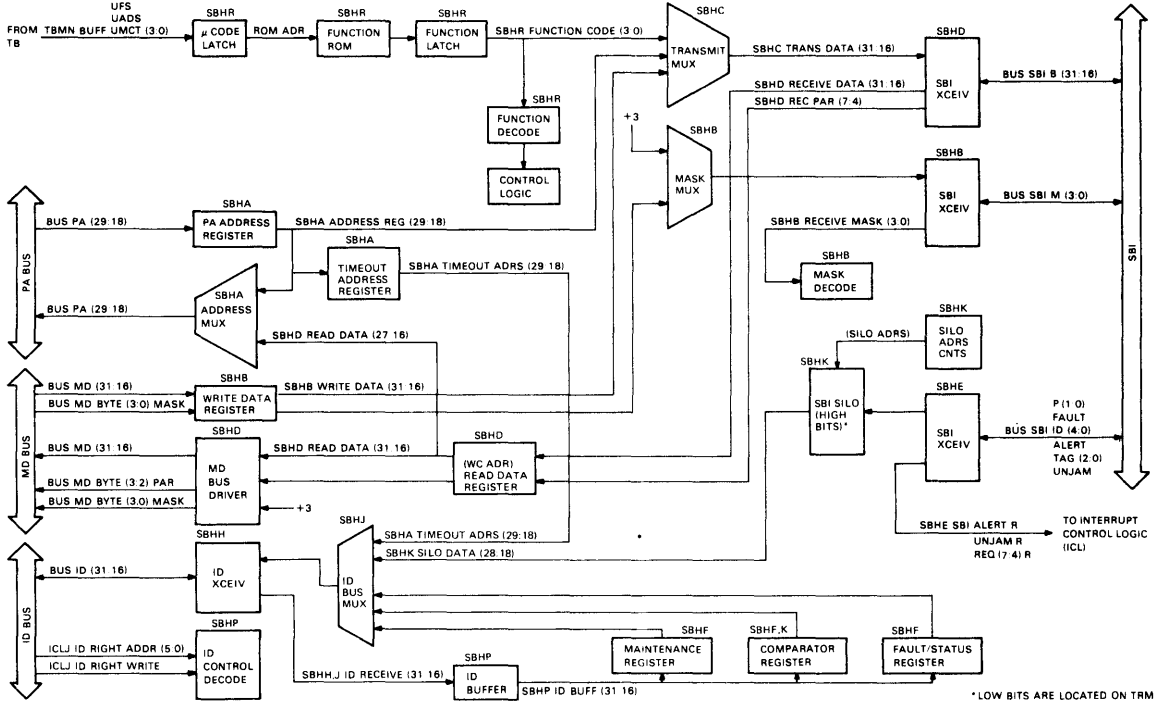
KA780 CPU BLOCK DIAGRAM



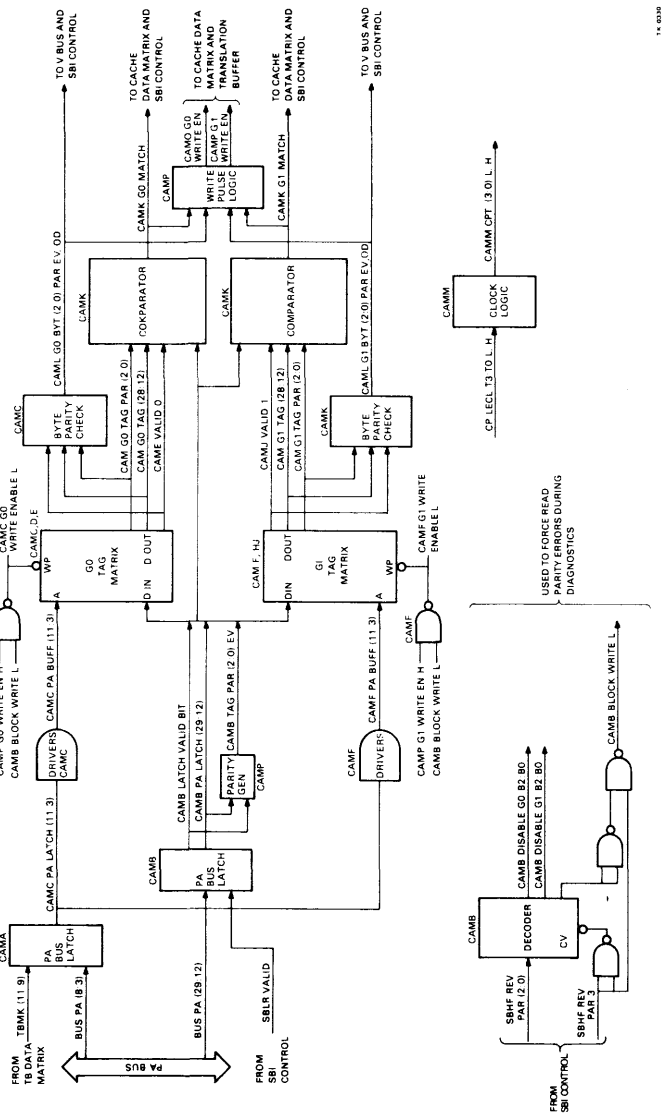
SBI CONTROL LOW BITS (SBL) BLOCK DIAGRAM



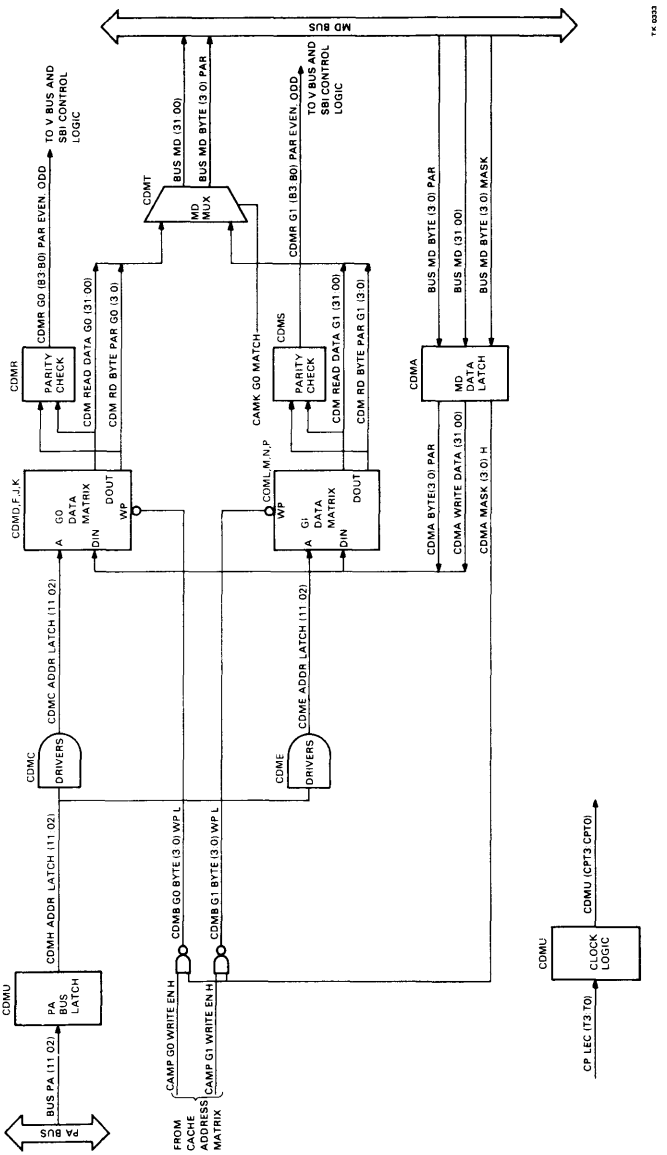
SBI CONTROL HIGH BITS (SBH) BLOCK DIAGRAM



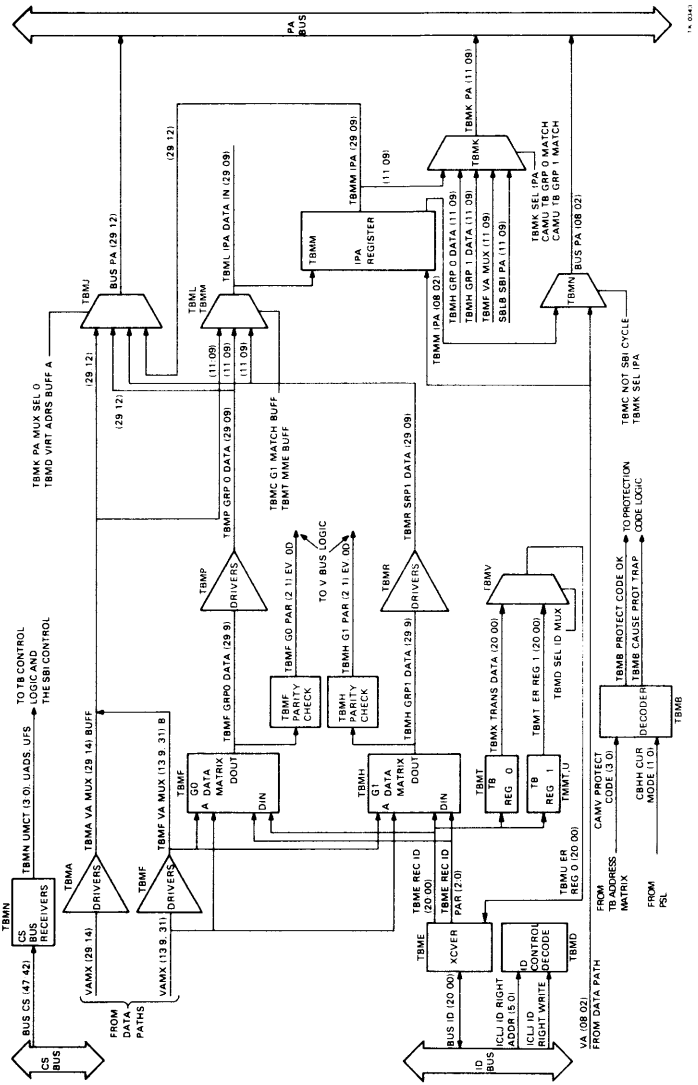
CACHE ADDRESS MATRIX BLOCK DIAGRAM



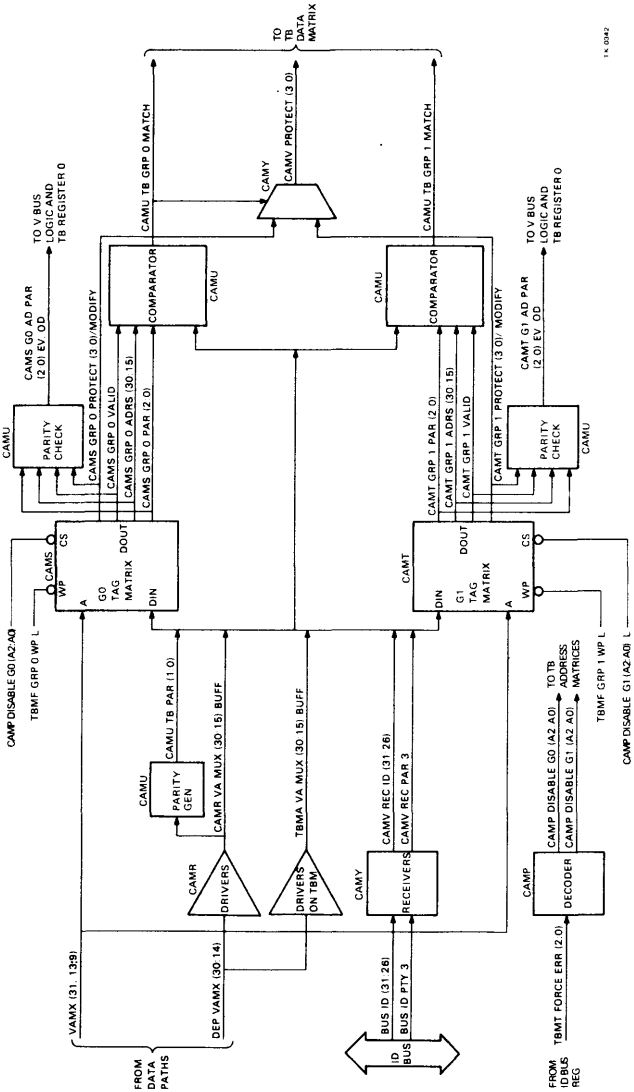
CACHE DATA MATRIX BLOCK DIAGRAM



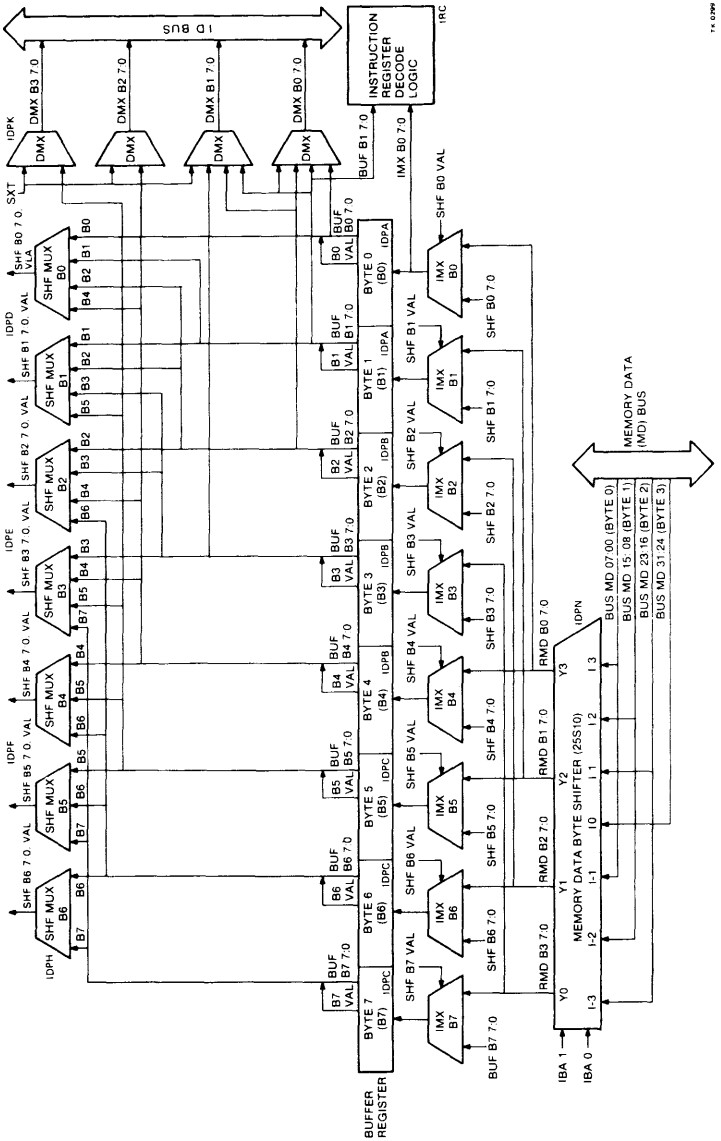
TRANSLATION-BUFFER DATA-MATRIX BLOCK DIAGRAM



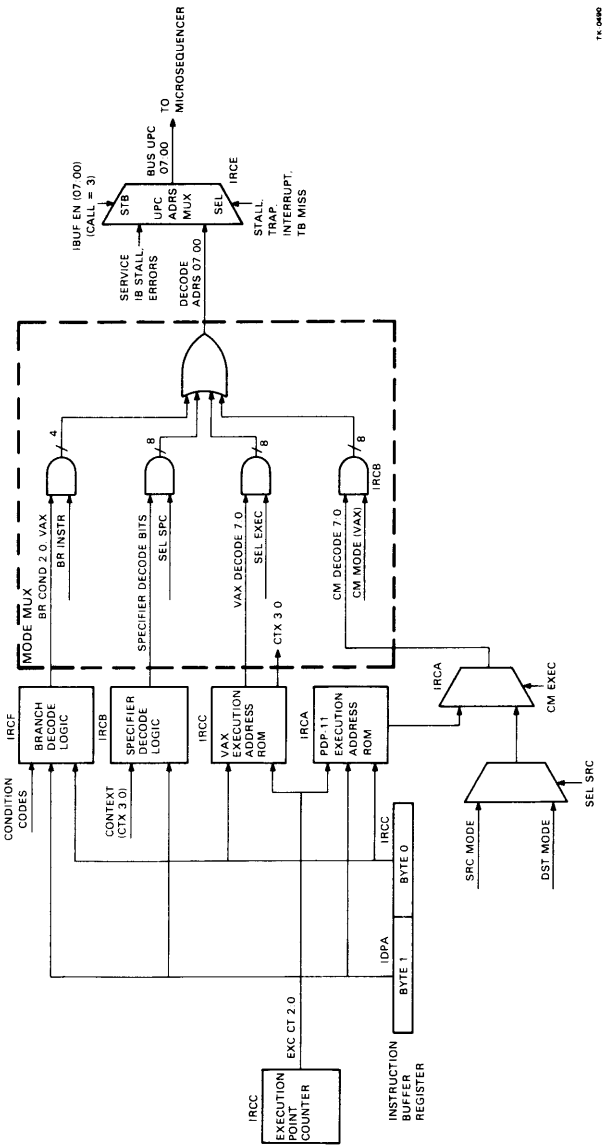
TRANSLATION-BUFFER ADDRESS-MATRIX BLOCK DIAGRAM

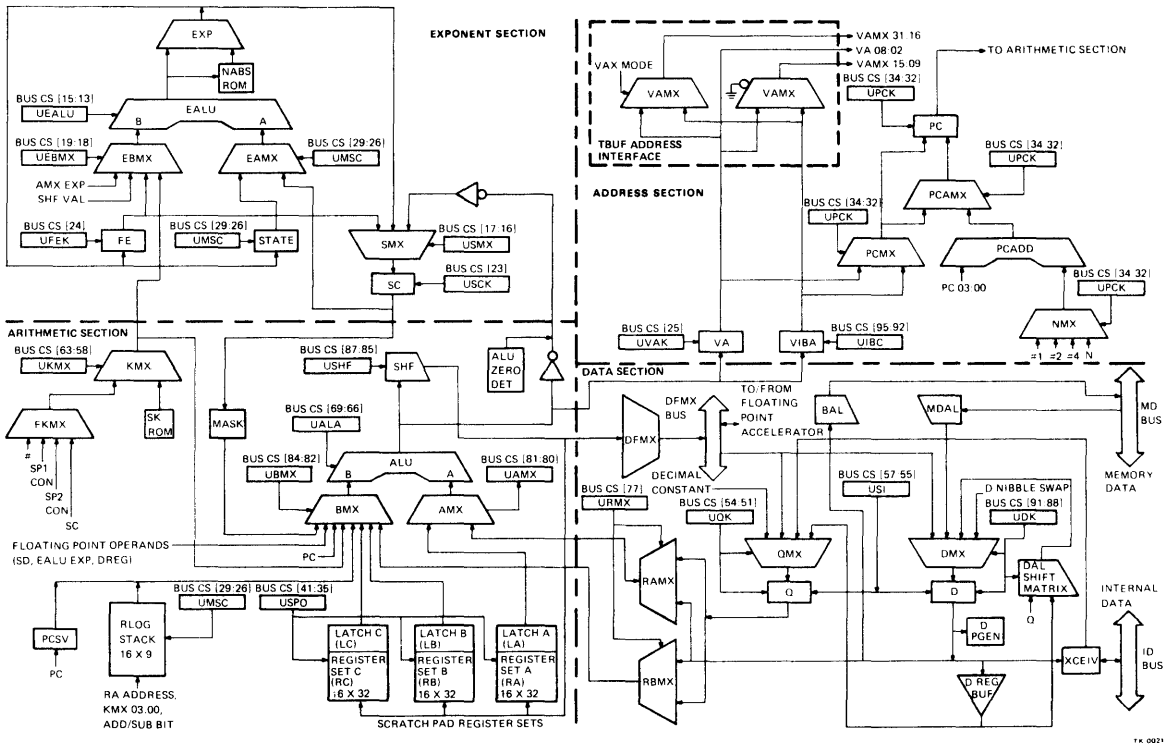


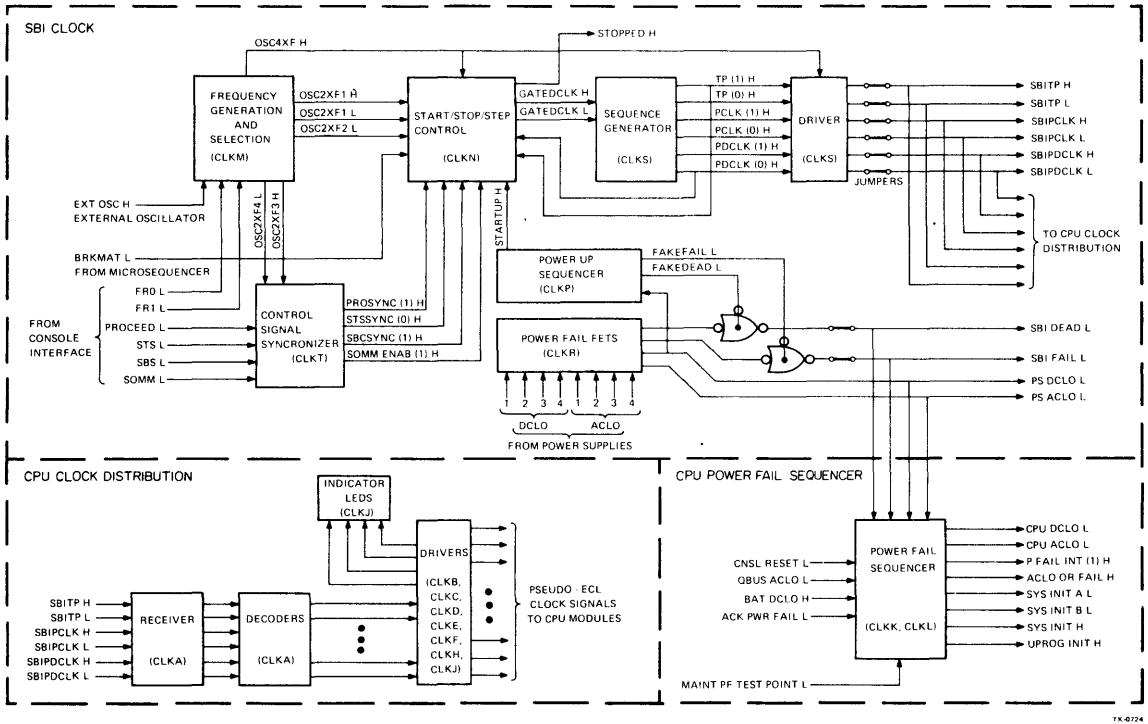
INSTRUCTION BUFFER BLOCK DIAGRAM



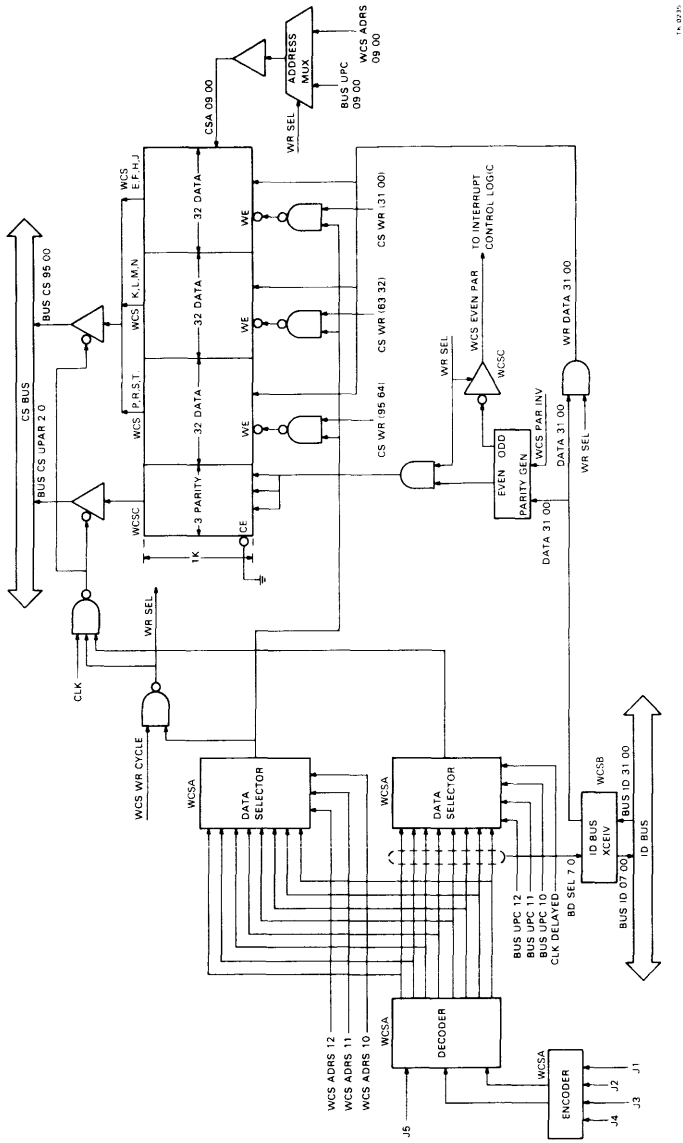
INSTRUCTION DECODE BLOCK DIAGRAM



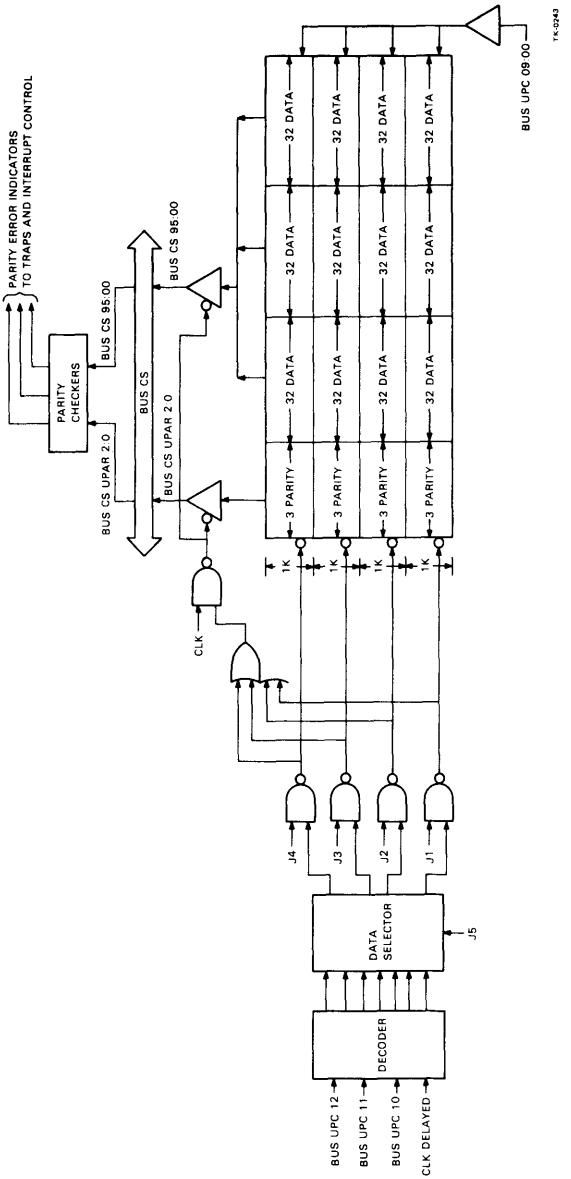

EXPONENT SECTION
EXP
NABS ROM
BUS CS [15:13]
UEALU
EALU
B
A
BUS CS [19:18]
UEBMX
EBMX
EAMX
BUS CS [29:26]
UMSC
AMX EXP
SHF VAL
BUS CS [24]
UFEK
FE
BUS CS [29:26]
UMSC
STATE
SMX
BUS CS [17:16]
USMX
SC
BUS CS [23]
USCK
ARITHMETIC SECTION
BUS CS [63:58]
UKMX
KMX
BUS CS [87:85]
USHF
SHF
ALU ZERO DET
FKMX
SK ROM
MASK
BUS CS [69:66]
UALA
SP1 CON
SP2 CON
SC
BUS CS [84:82]
UBMX
ALU
BUS CS [81:80]
UAMX
BMX
AMX
FLOATING POINT OPERANDS (SD, EALU EXP, DREG)
PC
BUS CS [29:26]
UMSC
BUS CS [41:35]
USPO
PCSV
PC
RLOG STACK 16 X 9
RA ADDRESS, KMX 03:00, ADD/SUB BIT
LATCH C (LC) REGISTER SET C (RC) 16 X 32
LATCH B (LB) REGISTER SET B (RB) 16 X 32
LATCH A (LA) REGISTER SET A (RA) 16 X 32
SCRATCH PAD REGISTER SETS
VAX MODE
VAMX
VAMX
TBUF ADDRESS INTERFACE
ADDRESS SECTION
VAMX 31:16
VA 08:02
VAMX 15:09
BUS CS [34:32]
UPCK
TO ARITHMETIC SECTION
PC
BUS CS [34:32]
UPCK
PCAMX
BUS CS [34:32]
UPCK
PCMX
PCADD
PC 03:00
BUS CS [34:32]
UPCK
NMX
=1 =2 =4 N
BUS CS [25]
UVAK
VA
VIBA
BUS CS [95:92]
UIBC
DATA SECTION
DFMX
DFMX BUS
TO/FROM FLOATING POINT ACCELERATOR
DECIMAL CONSTANT
BUS CS [77]
URMX
BUS CS [54:51]
UOK
BAL
MDAL
MD BUS
MEMORY DATA
BUS CS [57:55]
USI
D NIBBLE SWAP
BUS CS [91:88]
UDK
RAMX
RBMX
QMX
Q
DMX
D
DAL SHIFT MATRIX
Q
INTERNAL DATA
D PGEN
D REG BUF
XCEIV
ID BUS
TK 0021

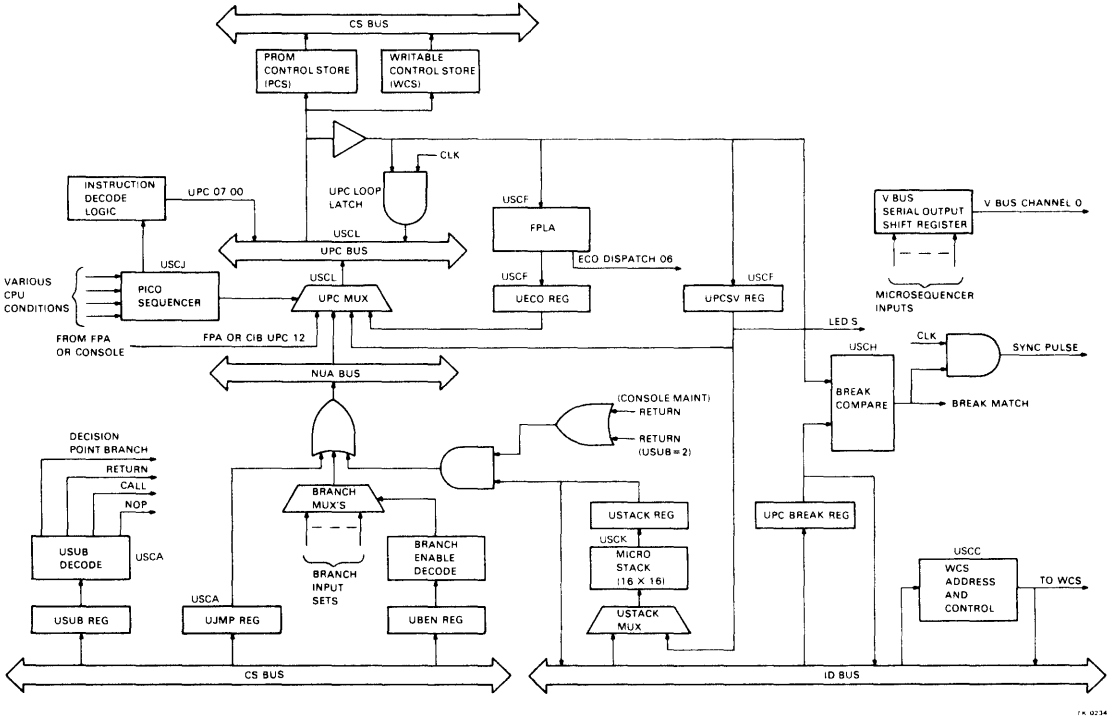

SBI CLOCK
OSC4XF H
STOPPED H
FREQUENCY GENERATION AND SELECTION (CLKM)
OSC2XF1 H
OSC2XF1 L
OSC2XF2 L
START/STOP/STEP CONTROL (CLKN)
GATEDCLK H
GATEDCLK L
SEQUENCE GENERATOR (CLKS)
TP (1) H
TP (0) H
PCLK (1) H
PCLK (0) H
PDCLK (1) H
PDCLK (0) H
DRIVER (CLKS)
JUMPERS
SBITP H
SBITP L
SBIPCLK H
SBIPCLK L
SBIPDCLK H
SBIPDCLK L
TO CPU CLOCK DISTRIBUTION
EXT OSC H
EXTERNAL OSCILLATOR
OSC2XF4 L
OSC2XF3 H
STARTUP H
BRKMAT L
FROM MICROSEQUENCER
FR0 L
FR1 L
FROM CONSOLE INTERFACE
PROCEED L
STS L
SBS L
SOMM L
CONTROL SIGNAL SYNCRONIZER (CLKT)
PROSYNC (1) H
STSSYNC (0) H
SBCSYNC (1) H
SOMM ENAB (1) H
POWER UP SEQUENCER (CLKP)
FAKEFAIL L
FAKEDEAD L
POWER FAIL FETS (CLKR)
SBI DEAD L
SBI FAIL L
PS DCLO L
PS ACLO L
1 2 3 4 1 2 3 4
DCLO
ACLO
FROM POWER SUPPLIES
CPU CLOCK DISTRIBUTION
INDICATOR LEDS (CLKJ)
DRIVERS (CLKB, CLKC, CLKD, CLKE, CLKF, CLKH, CLKJ)
SBITP H
SBITP L
SBIPCLK H
SBIPCLK L
SBIPDCLK H
SBIPDCLK L
RECEIVER (CLKA)
DECODERS (CLKA)
PSEUDO - ECL CLOCK SIGNALS TO CPU MODULES
CPU POWER FAIL SEQUENCER
CNSL RESET L
QBUS ACLO L
BAT DCLO H
ACK PWR FAIL L
POWER FAIL SEQUENCER (CLKK, CLKL)
CPU DCLO L
CPU ACLO L
P FAIL INT (1) H
ACLO OR FAIL H
SYS INIT A L
SYS INIT B L
SYS INIT H
UPROG INIT H
MAINT PF TEST POINT L
TK-0724

WRITABLE CONTROL STORE (WCS) BLOCK DIAGRAM

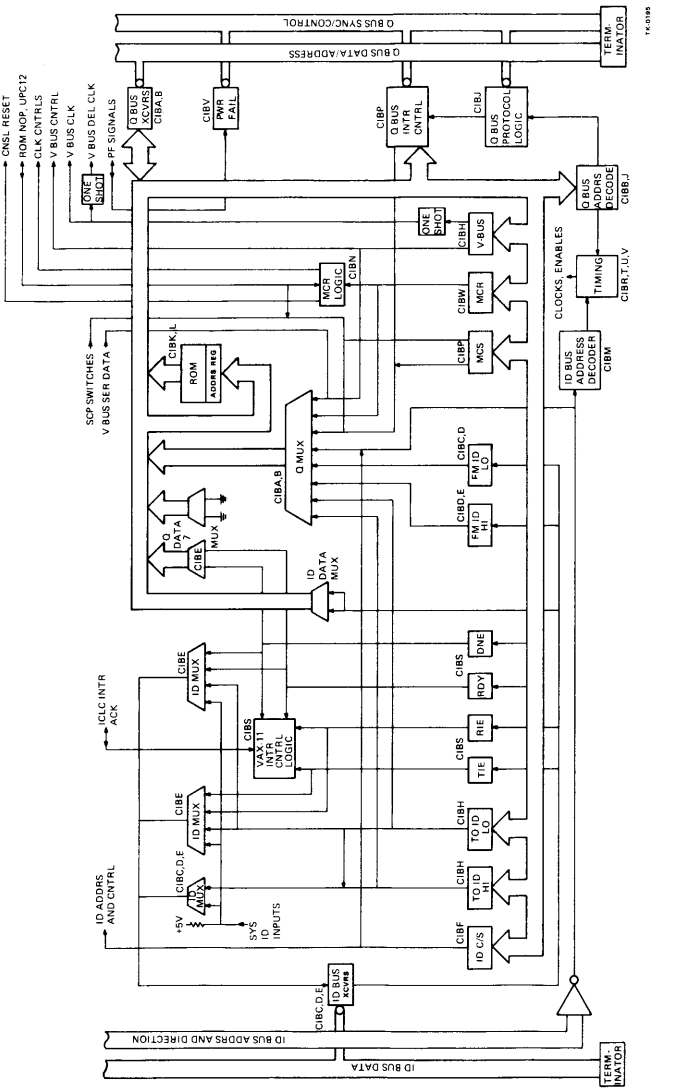


PROM CONTROL STORE (PCS) BLOCK DIAGRAM

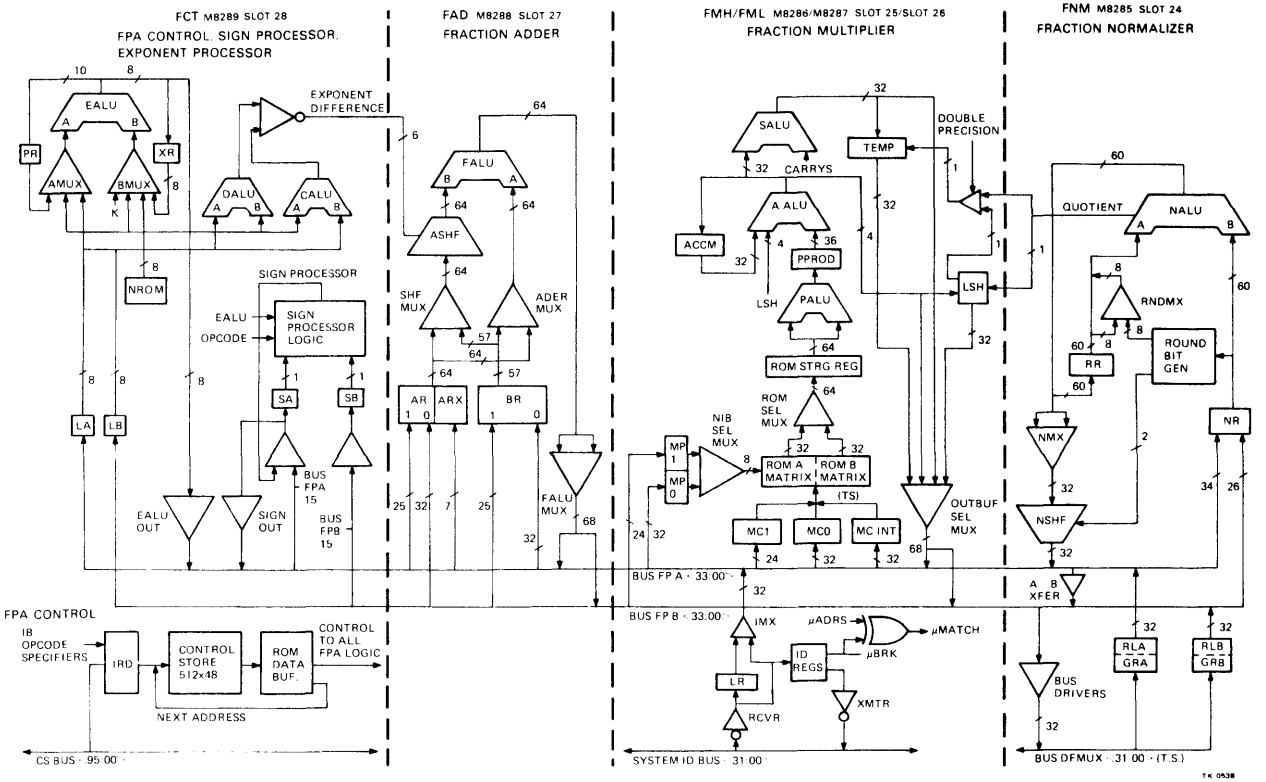



CS BUS
PROM CONTROL STORE (PCS)
WRITABLE CONTROL STORE (WCS)
CLK
INSTRUCTION DECODE LOGIC
UPC 07 00
UPC LOOP LATCH
USCF
FPLA
V BUS SERIAL OUTPUT SHIFT REGISTER
V BUS CHANNEL 0
USCL
UPC BUS
ECO DISPATCH 06
MICROSEQUENCER INPUTS
USCJ
VARIOUS CPU CONDITIONS
PICO SEQUENCER
UPC MUX
UECO REG
UPCSV REG
LED S
FROM FPA OR CONSOLE
FPA OR CIB UPC 12
USCH
CLK
SYNC PULSE
NUA BUS
BREAK COMPARE
BREAK MATCH
(CONSOLE MAINT) RETURN
RETURN (USUB = 2)
DECISION POINT BRANCH
RETURN
CALL
NOP
BRANCH MUX'S
USTACK REG
UPC BREAK REG
USCK
USUB DECODE
USCA
BRANCH INPUT SETS
BRANCH ENABLE DECODE
MICRO STACK (16 X 16)
USCC
WCS ADDRESS AND CONTROL
TO WCS
USCA
USUB REG
UJMP REG
UBEN REG
USTACK MUX
CS BUS
ID BUS

CONSOLE INTERFACE BOARD BLOCK DIAGRAM

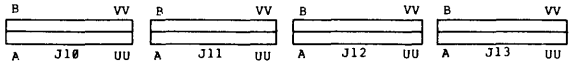


FLOATING-POINT ACCELERATOR (FPA) BLOCK DIAGRAM



KA780 TR AND SYSTEM ID REGISTER JUMPERS

b	d	f	j	l	n	r	t	v	x	z	bb	dd	ff	jj	ll	nn	rr	tt	vv
a	c	e	h	k	m	p	s	u	w	y	aa	cc	ee	hh	kk	mm	pp	ss	uu



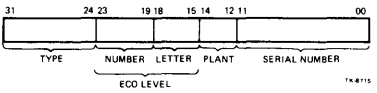
TR Arbitration Level

Signal Name	TR SEL 8	TR SEL 4	TR SEL 2	TR SEL 1	
TR#	J10 J	J10 F	J10 D	J10 B	Wire Wrap F02M2 to
1	--	--	--	--	F02C1
2	--	--	--	I	F02D1
3	--	--	I	--	F02E1
4	--	--	I	I	F02F2
5	--	I	--	--	F02H2
6	--	I	--	I	F02J1
7	--	I	I	--	F02J2
8	--	I	I	I	F02M1
9	I	--	--	--	F02N1
10	I	--	--	I	F02P1
11	I	--	I	--	F02P2
12	I	--	I	I	F02S2
13	I	I	--	--	F02T2
14	I	I	--	I	F02U1
15	I	I	I	--	F02U2
16	I	I	I	I	-----

System ID Register
Remove Jumper to Assert Bit

Bit	Jumper		Signal
0	J13	VV	
1	J13	TT	
2	J13	RR	
3	J13	NN	
4	J13	LL	SYSTEM
5	J13	JJ	SERIAL
6	J13	FF	NUMBER
7	J13	DD	
8	J13	BB	
9	J13	Z	
10	J13	X	
11	J13	V	
12	J13	T	MFG
13	J13	R	PLANT
14	J13	N	ID
15	J13	L	
16	J13	J	
17	J13	F	
18	J13	D	CPU
19	J13	B	CLUSTER
20	J12	VV	ECO
21	J12	TT	LEVEL
22	J12	RR	
23	J12	NN	
24	J12	LL	
25	J12	JJ	SYSTEM
26	J12	FF	TYPE
27	J12	DD	01-11/780
28	J12	BB	02=?
29	J12	Z	03=?
30	J12	X	
31	J12	V	

SYSTEM IDENTIFICATION REGISTER (SID)



KA780 WCS JUMPERS

	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	--
1-2K	--	I	I	--	I	--	I	I	--	I
2-3K	--	I	--	I	I	--	I	--	I	I
3-4K	--	--	I	I	I	--	--	I	I	I
4-5K	I	I	I	I	--	I	I	I	I	--
5-6K	I	I	I	--	I	I	I	I	--	I
6-7K	I	I	--	I	I	I	I	--	I	I
7-8K	I	--	I	I	I	I	--	I	I	I

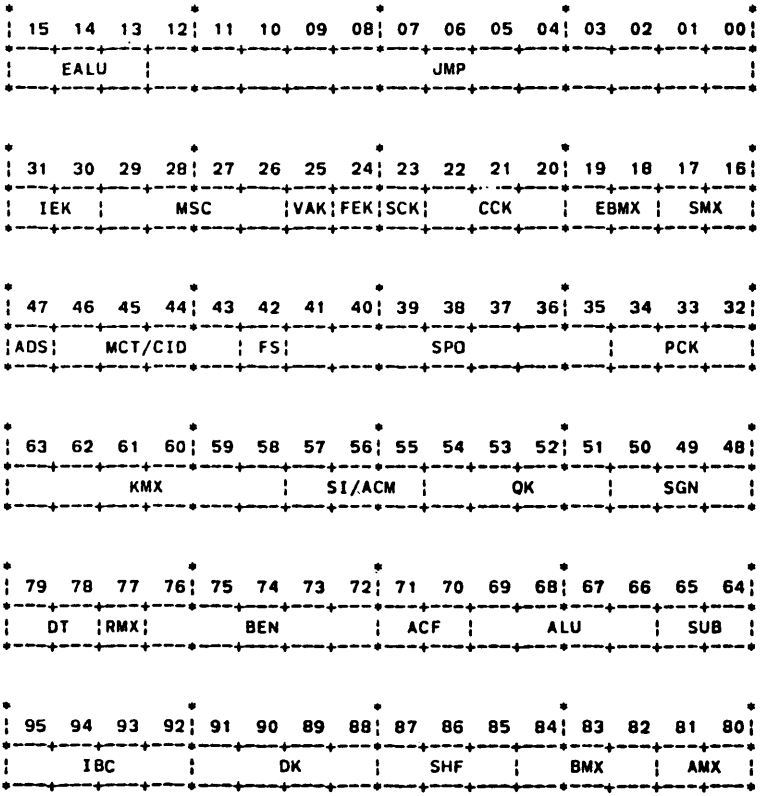
	PCS Slot 22 J12				
	L	J	F	D	B
0-4K	I	--	--	--	--

NOTES:

1. Addresses 0-4K are reserved for PCS.
2. M8233 starting addresses are in 1K increments.
3. M8238 starting addresses are in 2K increments and begin on even boundaries only.

CHAPTER 3
MICROCODE

CONTROL STORE FIELD MAP



MICROCODE ROUTINES THAT SUPPORT CONSOLE SOFTWARE STARTING ADDRESSES

```
        "CONSOLE MICRO-CODE"


        ;MICRO-CODE ROUTINES TO SUPPORT CONSOLE SOFTWARE.
        ;ROUTINES EXPECT DATA IN RXDB, AND IN ID[T1],ID[T2]
        ;AND THEY RETURN DATA IN TXDB, STATUS IN ID[D.SV],
        ;AND ADDITIONAL INFORMATION IN ID[T3].
        ;PC IS USED WHENEVER R15 IS REFERENCED.
        ;NO EFFORT IS MADE TO SAVE INTERNAL REGISTERS,

        ;INFORMATION AND PARAMETERS NEEDED FROM THE CONSOLE,
        ;ARE LOADED IN ID[RXDB] AND ID[T1],ID[T2].
        ;RESULTING DATA IS LOADED IN ID[TXDB] AND ID[T3],
        ;AND STATUS INFORMATION IS LOADED IN ID[D.SV].

;ROUTINE:       START-ADDRESS:  PARAMETERS:(* MEANS SUPPLIED BY CONSOLE)
;-------------------------------------------------------------------------
;EXAMINE MEMORY 120             ID[T1]=BYTE/WORD/LONG-PARAMETER *
;                               ID[RXDB]=VIRTUAL ADDRESS *
;                               ID[TXDB]=MEMORY DATA
;                               ID[T3]=PHYSICAL ADDRESS
;                               ID[D.SV]=STATUS-CODE
;-------------------------------------------------------------------------
;DEPOSIT MEMORY 121             ID[T1]=BYTE/WORD/LONG-PARAMETER *
;                               ID[RXDB]=VIRTUAL ADDRESS *
;                               ID[T2]=MEMORY DATA *
;                               ID[TXDB]=PHYSICAL ADDRESS
;                               ID[D.SV]=STATUS-CODE
;-------------------------------------------------------------------------
;EXAM.GEN.REG.  122             ID[RXDB]=REGISTER NUMBER *
;                               ID[TXDB]=REGISTER DATA
;-------------------------------------------------------------------------
;DEPO.GEN.REG.  123             ID[RXDB]=REGISTER NUMBER *
;                               ID[T2]=REGISTER DATA *
;-------------------------------------------------------------------------
;EXAM.PROC.REG. 124             ID[RXDB]=REGISTER NUMBER *
;                               ID[TXDB]=REGISTER DATA
;-------------------------------------------------------------------------
;DEP.PROC.REG.  125             ID[RXDB]=REGISTER NUMBER *
;                               ID[T2]=REGISTER DATA *
;-------------------------------------------------------------------------
;CONTINUE       127
;-------------------------------------------------------------------------
;QUAD-CLEAR     129             ID[RXDB]=QUAD-ADDRESS *
;-------------------------------------------------------------------------
;SBI-UNJAM      12A
;-------------------------------------------------------------------------
```

BEN	Name	UPC<Ø2>	UCP<Ø1>	UPC<ØØ>
Ø	NOP	Ø	Ø	Ø
1	Z	Ø	Ø	ALU Z
2	ROR	LA<Ø1>	PSL<C>	LA<ØØ>
3	C31	Ø	ALU C31	Ø
4	IRC.ROM	IRC.ROM	IRC.ROM	IRC.ROM
5	IB.O	IB.O		
6	ACCelerator	ACC UB2	ACC UB1	ACC UBØ
7				
8 VAX	DATA TYPE Ø = Normal 2 = Field Src 1 = Q + D 3= Addr Src	Read + Modify	ASRC + VSRC	ASRC + Q + D
8 -11	END DP1	Read + Modify	D Class	J Class + DM27
9 VAX	IR2-1	Ø	IR<2>	IR<1>
9 -11	PC Modes	Ø	SM or DM 47 + 57	Dst R .eq. PC
A	REI	Mode .lt. ASTLVL		
B	IB TEST Ø = TB Miss 2 = Stall 1 = Error 3 = Data OK	Ø	IB running	IB ERROR + DATA VALID
C	MUL	SC.ne.Ø	D<Ø1>	D<ØØ>
D	SIGNS	Q<31>	D.ne.Ø	D<31>
E	INTERRUPT	AC Low	Internal Interrupt	Interrupt Request
F	Decimal	Ø	D<7:Ø> 3Ø-39	D<3:Ø>= Ø8 + ØD

BEN	Name	UPC<Ø3>	UPC<Ø2>	UCP<Ø1>	UPC<Ø0>
1Ø	uTrap Vector	uVECT<3>	uVECT<2>	uVECT<1>	uVECT<Ø>
11	Last Reference	-PSL<FPD>	Nested Error	Wr Chk* -Intlk	-Read+ Intlk
12	EALU CC	EALU N	EALU Z	SC.ne.Ø	Sign Src
13					
14	SC Ø = Zero 1 = Neg 2 = 1-31 3 = .gt.31	Ø	SC<9:8> .ne.Ø	SC.gt.Ø	SC<9:5> .ne.Ø
15	ALU1-Ø (previous cycle)	Rlog Empty	ALU<1:Ø> .eq.Ø	ALU<Ø1>	ALU<Ø0>
16	STATE7-4	STATE<7>	STATE<6>	STATE<5>	STATE<4>
17	STATE3-Ø	STATE<3>	STATE<2>	STATE<1>	STATE<Ø>
18	D Bytes	D<31:24> .ne.Ø	D<23:16> .ne.Ø	D<15:8> .ne.Ø	D<7:Ø> .ne.Ø
19	D3-Ø	D<Ø3>	D<Ø2>	D<Ø1>	D<Ø0>
1A	PSL CC	PSL<N>	PSL<Z>	PSL<V>	PSL<C>
1B	ALU CC	ALU N	ALU Z	IR<Ø>	ALU C31

BEN	Name	UPC<Ø4>	UPC<Ø3>	UPC<Ø2>	UPC<Ø1>	UPC<Ø0>
1C	PSL Mode	-VAMX<31>	-VAMX<3Ø>	-Console Mode	-PSL<IS>* -PSL<CM>	Kernel Mode
1D	Translation Test	PTE -Valid	Data Aligned	Ø	TB Miss + Access Viol.	TB Miss + 1st Modify
1E						
1F						

MICROTRAP VECTORS

Number	Function
100	System Initialization
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 or Error Confirm
10E	Odd Address Error
10F	Control Store Parity Error

MICROCODE MEMORY CONTROL FUNCTIONS

```
                        ---  trap on  ---
          |  |  F|   |  |T A          D       S|    Y = utrap on condition
          |  |  U|  C|  |B C  X  A  T D  T    B|    * = utrap on condition unless MSC/
          |  |  N|  H|S |M C  P  L  B D  B C  I|        SECOND.REF or RETRY.NO.TRAP
A         |V|  C|  E|A |I E  A  I  U A  P P  E|    N = do not utrap on condition
D  MCT  F|/|  T|  C|V|S S  G  G  F D  A A  R|    - = hardware behaviour undefined.
S 3210  S|P|  N|  K|E|S S  E  N  M R  R R  R|        ucode must prevent condition
--------+-+---+---+-+---------------+--------------------
0 0000  0|V|   |  R|N|N N  N  N  N N  Y N  N| TEST.RCHK
0 0001  0|V|   |   |N|N N  N  N  N N  N N  N| MEM.NOP
0 0010  0|V|   |  W|N|N N  N  N  N Y  N N  N| TEST.WCHK
0 0011  0|  |   |   | |                      |

0 0100  0|  |   |   | |                      |
0 0101  0|V|  W|   |N|Y N  -  -  N -  Y N  Y| WRITE.V.NOCHK
0 0110  0|V|  W|  W|Y|Y Y  *  *  Y Y  Y N  Y| WRITE.V.WCHK
0 0111  0|V|IW|   |N|-  -  -  -  - -  Y N  Y| LOCKWRITE.V.XCHK

0 1000  0|V|  R|  R|Y|Y Y  *  *  N Y  Y Y  Y| READ.V.RCHK
0 1001  0|V|  R|   |N|Y N  -  -  N -  Y Y  Y| READ.V.NOCHK
0 1010  0|V|  R|  W|Y|Y Y  *  *  Y Y  Y Y  Y| READ.V.WCHK
0 1011  0|V|  R|IB|Y|Y Y  Y  Y  Y Y  Y Y  Y| READ.V.IBCHK

0 1100  0|V|  R|  R|N|N N  N  N  N Y  N N  N| READ.V.NEWPC
0 1101  0|V|IR|   |N|Y N  -  -  N -  Y Y  Y| LOCKREAD.V.NOCHK
0 1110  0|V|IR|  W|Y|Y Y  -  -  Y -  Y Y  Y| LOCKREAD.V.WCHK
0 1111  0|  |   |   | |                      |

1 0000  0|  |HOLD |N|N N  N  N  N N  N N  N| SBI.HOLD
1 0001  0|  |UNJAM|N|N N  N  N  N N  N N  N| SBI.HOLD+UNJAM
1 0010  0|P|INVAL|N|N N  N  N  N N  N N  N| INVALIDATE
1 0011  0|P|  VAL |N|N N  N  N  N N  N N  N| VALIDATE

1 0100  0|P|EXTwR|N|N N  N  N  N N  N N  Y| EXTWRITE.P
1 0101  0|P|   W  |N|N N  N  N  N N  N N  Y| WRITE.P
1 0110  0|  |      | |                      |
1 0111  0|P|  IW  |N|N N  N  N  N N  N N  Y| LOCKWRITE.P

1 1000  0|  |      | |                      |
1 1001  0|P|  R   |N|N N  N  N  N N  N Y  Y| READ.P
1 1010  0|  |      | |                      |
1 1011  0|P| ISR  |N|N N  N  N  N N  N N  Y| READ.INT.SUM

1 1100  0|  |      | |                      |
1 1101  0|P|  IR  |N|N N  N  N  N N  N Y  Y| LOCKREAD.P
1 1110  0|  |      | |                      |
1 1111  0|I|  R   |N|N N  N  N  N N  N N  N| ALLOW.IB.READ

0 XXXX  1|  |      | |N N  N  N  N N  N N  N| NO MEMORY OPERATION
1 XXXX  1|I|  R   |N|N N  N  N  N N  N N  N| DEFAULT: ALLOW IB READ

Abort Ref on Trap? A A A   A A A R A (A=any, R=read)
```

Hex Value	USPO Field BUS CS 41	40	39	38	37	36	35	Scratch Pad Operation
00-05	0	0	0	0	X	X	X	No operation
06	0	0	0	0	1	1	0	Load LC ;Address = SC03:00
07	0	0	0	0	1	1	1	Write RC, RB ;Address = SC03:00
08-0F	0	0	0	1	A	C	N	Load LA, LB ;Address determined by ACN value
10-17	0	0	1	0	R	N		Load LA ;Address = General Register (R0-R7)
18-1F	0	0	1	1	A	C	N	Write RA, RB ;Address determined by ACN value
20-2F	0	1	0		R	N		Load LC ;Address = Temporary Register (T0-TF)
30-3F	0	1	1		R	N		Write RC ;Address = Temporary Register (T0-TF)
40-4F	1	0	0		R	N		Load LA, LB ;Address = General Register (R0-RF)
50-5F	1	0	1		R	N		Write RA, RB ;Address = General Register (R0-RF)
60-6F	1	1	0		R	N		Load LA, LB ;Address = General Register (R1) and Write RC ;Address = Temporary Register (T0-TF)
70-7F	1	1	1		R	N		Load LC ;Address = Temporary Register (T0-TF) Write RA, RB ;Address = General Register (R1)

```
.TOC    "       Machine definition    : ACF, ACM, ADS, ALU, AMX"

ACF/=<71:70>,.DEFAULT=0          ;ACCELERATOR CONTROL
        NOP=0
        SYNC=1
        TRAP=2
        CONTROL=3                ;ACCELLERATOR-DEPENDENT CONTROL FUNCTION

ACM/=<57:55>                     ;ACCELERATOR MISCELLANEOUS CONTROL
        PWR.UP=0
        ABORT=1                  ;RETURN ACCEL TO MONITORING IRD
        POLY.DONE=6

ADS/=<47:47>                     ;ADDRESS SELECT
        VA=0
        IBA=1

ALU/=<69:66>,.DEFAULT=0F         ;ALU CONTROL FUNCTIONS
        A-B=00
        A-B.RLOG=01
        A-B-1=02
        INST.DEP=03                      ;INSTRUCTION DEPENDENT
        A+B+1=04
        A+B=05
        A+B.RLOG=06
        ORNOT=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/=<81:80>                             ;AMX TO ALU
        LA=0
        RAMX=1
        RAMX.SXT=2                       ;RAMX SIGN EXTENDED ACCORDING TO DT
        RAMX.0XT=3                       ;RAMX ZERO EXTENDED. 0XT(L)=0
```

```
.TOC    "       Machine definition      : BEN, BMX"

BEN/=<76:72>,,DEFAULT=0         ;BRANCH ENABLE
        NOP=0                   ;NO BRANCH
        Z=1                     ; ALU Z
        ROR=2                   ;LA<1>, PSL<C>, LA<0>
        C31=3                   ; ALU C31, 0
        IRC.ROM=4               ;OUTPUT OF EXTENDED IRC-ROM
        IB.0=5                  ;IB 0 READY ?
        ACCEL=6                 ;CODE FROM ACCELERATOR
        DATA.TYPE=8             ;(VAX MODE) *, ASRC+VSRC, ASRC+0+D
                                ;   0--NORMAL, 1--QUAD OR DOUBLE
                                ;   2--FIELD,  3--ADDRESS
        END.DP1=8               ;(-11 MODE) *, 0 CLASS, J CLASS+D=27
        IR2-1=9                 ;(VAX MODE) *, IR<2:1>
        PC.MODES=9              ;(-11 MODE) *, SM47+SM57+DM47+DM57, DST R=PC
        REI=0A                  ;(VAX MODE) MODE.LSS.ASTLVL, *, *
        SRC.PC=0A               ;(-11 MODE) SRC R=PC
        IB.TEST=0B              ; 0--TB MISS, 1--ERROR
                                ; 2--STALL, 3--DATA OK
        MUL=0C                  ;SC.NE.0, D<1:0>
        SIGNS=0D                ;Q<31>, D.NE.0, D<31>
        INTERRUPT=0E            ;AC LOW, INTERNAL INTERRUPT, INT REQ
        DECIMAL=0F              ;0, D BYTE 0 VALID DIGIT, D2-0 NEG SIGN
;       UTRAP=10                ;MICROTRAP DISPATCH VECTOR
        LAST.REF=11             ;-FPD, NESTED ERROR, LOW TWO BITS:
                                ; 0--READ INTERLOCK, 1--READ, READ CHK
                                ; 2--WRITE, 3--READ, WRITE CHK
        EALU=12                 ;EALU N, EALU Z, SC.NEQ.0, SS
        SC=14                   ;SC<9:8>.NE.0, SC.GT.0, SC<9:5>.NE.0
        ALU1-0=15               ;RLOG EMPTY, ALU<1:0>=0, ALU<1>, ALU<0>
                                ; (ALU BITS FROM PREVIOUS STATE)
        STATE7-4=16             ;STATE <7:4>
        STATE3-0=17             ;STATE <3:0>
        D.BYTES=18              ;BYTES 3, 2, 1, 0 OF D.NE.0
        D3-0=19                 ;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             ;-VA<31:30>, -CONSOLE, IS+CM, KERNEL
        TB.TEST=1D              ;PTE VALID, ALIGNED, QUAD, +
                                ; 0--TRANSLATION OK, 1--WR CHK AND M=0
                                ; 2--ACCESS VIOLATION, 3--TB MISS

BMX/=<84:82>                    ;BMX TO ALU
        MASK=0                  ;A 0 IN THE BIT SELECTED BY SC<4:0>
        PC.OR.LB=1              ;LB UNLESS R=PC, THEN PC
        PACKED.FL=2             ;PACKED FLOATING
        LB=3
        LC=4
        PC=5
        KMX=6
        RBMX=7                  ;D OR Q
```

```
.TOC    "       Machine definition      : CCK, CID, DK, DT"

CCK/=<22:20>,.DEFAULT=0 ;CONDITION CODES
                                        ;Note : * = RESERVED OPERATION, "ALU SIGN" AND "AMX SIGN" ARE SIZE DEPENDENT
                                        ;-----------------------------------------------------------------------------------------+
                                        ;           NATIVE MODE PSL                 |          COMPATIBILITY MODE PSL             |
                                        ;-----------+-------------------+---+---+---+-----------+-------------------+---+---------+
                                        ;     N     |         Z         | V | C |   N       |         Z         | V |    C    |
                                        ;-----------+-------------------+---+---+-----------+-------------------+---+---------+
        NOP=0                           ;     N     |         Z         | V | C |   N       |         Z         | V |    C    |
        LOAD.UBCC=1                     ;     N     |         Z         | V | C |   N       |         Z         | V |    C    |
        SET.V=2         ,.VALIDITY=<V1>;     N     |         Z         | 1 | C |   *       |         *         | * |    *    |
        N_AMX.Z_1ST.VC_VC=3             ; AMX SIGN  | Z.and.(ALU.eq.0)  | V | C | AMX SIGN  | Z.and.(ALU.eq.0)  | V |    C    |
        ROR=4           ,.VALIDITY=<V0>;     *     |         *         | * | * | ALU SIGN  |      ALU.eq.0     | 0 | AMX<0>  |
        NZ_ALU.VC_0=5                   ; ALU SIGN  |      ALU.eq.0     | 0 | 0 | ALU SIGN  |      ALU.eq.0     | 0 |    0    |
        NZ_ALU.VC_VC=6  ,.VALIDITY=<V1>; ALU SIGN  |      ALU.eq.0     | V | C |           |                   |   |         |
        C_AMX0=6        ,.VALIDITY=<V0>;           |                   |   |   |     N     |         Z         | V | AMX<0>  |
        INST.DEP=7                      ;                        Instruction dependent                                          |
                                        ;-----------+-------------------+---+---+-----------+-------------------+---+---------+

CID/=<45:42>                    ;CONSOLE & ID BUS CONTROL IF FS/1
        NOP=1                   ;DEFAULT, ALLOW AUTO IB READ
        ACK=5                   ;SET CONSOLE ACKNOWLEGE FLAG
        CONT=7                  ;CLEAR CONSOLE MODE
        READ.SC=9               ;READ ID BUS REG SELECTED BY SC
        READ.KMX=0B             ;READ ID BUS REG SELECTED BY UKMX
        WRITE.SC=0D             ;WRITE REG SELECTED BY SC
        WRITE.KMX=0F            ;WRITE REG SELECTED BY UKMX

DK/=<91:88>,.DEFAULT=0
        NOP=0                   ;DEFAULT, HOLD
        LEFT2=1                 ;DOUBLE SHIFT LEFT
        RIGHT2=2                ;DOUBLE SHIFT RIGHT
        DIV=4                   ;IF NOT ALU CRY, SHIFT LEFT
                                ; ELSE LOAD FROM SHF
        LEFT=5                  ;SHIFT LEFT
        RIGHT=6                 ;SHIFT RIGHT
        SHF=8                   ;LOAD SHF MUX, INTEGER FORMAT
        SHF.FL=9                ;LOAD SHF MUX, UNPACKED FLOATING FORMAT
        ACCEL=0A                ;LOAD ACCELERATOR DATA FROM DF BUS
        BYTE.SWAP=0B            ;REFLECT BYTES AROUND BIT 16
        Q=0C                    ;LOAD Q THRU DAL
        DAL.SC=0D               ;LOAD DAL[SC]
        DAL.SV=0E               ;LOAD DAL[SHF VAL]
        CLR=0F                  ;LOAD ZEROS

DT/=<79:78>,.DEFAULT=0          ;DATA TYPE
                                ;CONTROLS AMX SIGN/ZERO EXTENDER, SHF AMOUNT,
                                ;CONDITION CODE SETTING, AND MEMORY REFERENCES
        LONG=0                  ;DEFAULT
        WORD=1
        BYTE=2
        INST.DEP=3              ;INSTRUCTION DEPENDENT -
                                ;ANY OF ABOVE, OR QUAD/DOUBLE
```

```
.TOC    "       Machine definition    : EALU, EBMX, FEK, FS, IEK, IBC"

EALU/=<15:13>                   ;EXPONENT ALU
        A=0
        OR=1
        ANDNOT=2
        B=3
        A+B=4
        A-B=5
        A+1=6
        ABS.A-B=7               ;-ABS(A-B)

EBMX/=<19:18>                   ;EBMX TO EALU
        FE=0                    ;DEFAULT
        MX=1
        AMX.EXP=2
        SHF.VAL=3               ;SHIFT VALUE

FEK/=<24:24>,.DEFAULT=0         ;FE REGISTER CONTROL
        NOP=0                   ;DEFAULT, HOLD
        LOAD=1

FS/=<42:42>                     ;FUNCTION SELECT FOR 43-46
        MCT=0                   ;ENABLE MEMORY CONTROL
        CID=1                   ;ENABLE ID BUS AND CONSOLE CONTROL

IEK/=<31:30>                    ;INTERRUPT AND EXCEPTION ACKNOWLEDGE
        NOP=0
        ISTR=1                  ;STROBE INTERRUPT REQUESTS
        IACK=2                  ;INTERRUPT ACKNOWLEDGE
        EACK=3                  ;EXCEPTION ACKNOWLEDGE

IBC/=<95:92>,.DEFAULT=0         ;IBUF CONTROL FUNCTIONS
        NOP=0                   ;DEFAULT
        STOP=1
        FLUSH=2                 ;FLUSH IB AND LOAD IBA
        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=0C                ;CLEAR BYTE 0 (VAX OPCODE)
        CLR.1=0D                ;CLEAR BYTE 1 (VAX SPECIFIER)
        CLR.0-3=0E              ;CLEAR BYTES 0-3 (-11 OP & DATA)
        CLR.1-5.COND=0F         ;CLEAR BYTES 1-5 CONDITIONALLY
                                ; IF THERE IS NO SPECIFIER EVALUATION,
                                ; CLEAR NOTHING.  IF A SELF-CONTAINED
                                ; SPECIFIER, CLEAR IT.  IF IMMEDIATE,
                                ; ABSOLUTE, OR DISPLACEMENT, CLEAR THE
                                ; ISTREAM LITERAL.
```

```
.TOC    "       Machine definition      : ID.ADDR, J"

ID.ADDR/=<63:58>                ;ID BUS ADDRESS
        IBUF=0                  ;RD     ;SPECIFIER/LITERAL DATA FROM IB
        DAY.TIME=1              ;RD+WR  ;CURRENT TIME OF DAY...
                                        ; MUST READ UNTIL STOPS CHANGING
        SYS.ID=3                ;RD     ;SYSTEM IDENTIFICATION
        RXCS=4                  ;RD+WR  ;CONSOLE RECIEVE CONTROL/STATUS
        RXDB=5                  ;RD     ;CONSOLE RECIEVE DATA BUFFER
                                        ; (TO-ID REGISTER)
        TXCS=6                  ;RD+WR  ;CONSOLE TRANSMIT CONTROL/STATUS
        TXDB=7                  ;WR     ;CONSOLE TRANSMIT DATA BUFFER
                                        ; (FROM-ID REGISTER)
;       DQ=8                            ;DATA PATH D/Q REGISTERS (MAINT ONLY)
        NXT.PER=9               ;WR     ;INTERVAL CLOCK NEXT PERIOD REGISTER
        CLK.CS=0A               ;RD+WR  ;INTERVAL CLOCK CONTROL/STATUS
        INTERVAL=0B             ;RD     ;CURRENT INTERVAL COUNT
        CES=0C                  ;RD+WR  ;CPU ERROR/STATUS
        VECTOR=0D               ;RD+WR  ;EXCEPTION & INTERRUPT CONTROL
        SIR=0E                  ;RD+WR  ;SOFTWARE INTERRUPT REGISTER
        PSL=0F                  ;RD+WR  ;PROCESSOR STATUS LONGWORD
        TBUF=10                         ;TRANSLATION BUFFER DATA
        TBER0=12                        ;TB ERROR/STATUS 0
        TBER1=13                        ;TB ERROR/STATUS 1
        ACC.0=14                        ;ACCELERATOR REGISTER #0
        ACC.1=15                        ;ACCELERATOR REGISTER #1
        ACC.2=16                        ;ACCELERATOR REGISTER #2
        ACC.CS=17                       ;ACCELERATOR CONTROL/STATUS
        SILO=18                         ;NEXT ITEM FROM SBI HISTORY
        SBI.ERR=19                      ;SBI ERROR REGISTER
        TIME.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             ;WR     ;WRITING WCS COUNTS ADDRESS
```

```
;ID BUS ADDRESSES CONTINUED.  ADDRESSES 24-3F ARE RAM LOCATIONS

        P0BR=24                         ;PROCESS SPACE 0 BASE REGISTER
        P1BR=25                         ;PROCESS SPACE 1 BASE REGISTER
        SBR=26                          ;SYSTEM SPACE BASE REGISTER
        KSP=28                          ;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
        D.SV=2E
        Q.SV=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                         ;PROCESS 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

.CREF

J/=<12:0>,.NEXTADDRESS                  ;NEXT MICRO WORD ADDRESS

.NOCREF
```

```
.TOC    "       Machine definition    : KMX"

KMX/=<63:58>            ;CONSTANTS OR # FROM FK
        .8=0            ;#8 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 (-11 MODE)
        ZERO=6          ; OR ZEROS (VAX MODE)
        SC=7            ;SC[9:0] FROM FK
                        ;8 - 3F: CONSTANTS (1 CYCLE SETUP IF ALU IN ARITH MODE)
                        ;DECIMAL VALUE OF CONSTANT
        .14=8           ;20     (AF,JL,MH)
        .A0=9           ;160    (AF,JL)
        .34=0A          ;52     (AF)
        .28=0B          ;40     (AF)
        .40=0C          ;64     (AF,JL,MH,TF)
        .50=0D          ;80     (AF,MH)
        .7FF0=0E        ;?      (TF) ****MACHINE-DEPENDENT!!!
        .EF=0F          ;239    (JL)
        .80=10          ;128    (AF,JL,MH,TF)
        .8000=11        ;-32768 (AF)
        .FF=12          ;255    (MH,TF)
        .FF00=13        ;-256   (MH,AF,JL)
        .1E=14          ;30     (AF)
        .3F=15          ;63     (MH,AF,TF)
        .7F=16          ;127
        .7=17           ;7      (AF,MH)
        .F=18           ;15     (MH,CM,AF,TF)
        .10=19          ;16     (MH,AF,JL,TF)
        .FFE8=1A        ;-24    (MH,TF)
        .FFF0=1B        ;-16    (CM,JL,TF,MH)
        .FFF8=1C        ;-8     (CM,TF,MH)
        .20=1D          ;32     (CM,JL,MH,TF)
        .30=1E          ;48     (CM,AF,MH,TF)
        .18=1F          ;24     (MH,AF,TF)
        .3FF=20         ;1023   (CM)
        .C=21           ;12     (CM,JL,TF,MH)
        .D=22           ;13     (TF)
        .1F=23          ;31     (AF,JL,MH,TF)
        .1F00=24        ;7936   (JL,MH)
        .B0=25          ;176    (MH)
        .E003=26        ;       (CM)
        .7C=27          ;124    (AF)
        .FFE0=28        ;-32    (JL)
        .60=29          ;96     (TF)
;       SPARE=2A
        .DFCF=2B        ;?      (JL)
        .4000=2C        ;?      (TF)****MACHINE-DEPENDENT!!
```

```
        .FFF1=2D        ;-15    (AF)****MACHINE-DEPENDENT!!
        .19=2E          ;25     (AF)
        .FFF9=2F        ;-7     (AF)
        .FFFF=30        ;-1     (MH,JL,TF)
        .88=31          ;136    (AF)
        .3030=32        ;?      (TF)
        .F0=33          ;240    (TF)
        .C0=34          ;192    (TF,MH)
        .6=35           ;6      (CM,JL,TF)
        .9=36           ;9      (CM)
        .FFF6=37        ;-10    (CM)
        .FFF5=38        ;-11    (CM)
        .1A=39          ;26     (CM,AF,TF)
        .24=3A          ;36     (CM,MH)
        .1B=3B          ;27     (CM,AF,TF)
        .FFFC=3C        ;-4     (CM,TF,MH)
        .A=3D           ;10     (AF,MH)
        .7E=3E          ;126    (AF,TF)
;       SPARE=3F
```

```
.TOC    "       Machine definition      : MCT, MSC"

MCT/=<47:42>,,DEFAULT=3E                ;MEMORY CONTROL
        TEST.RCHK=00                    ;TEST TBUF WITH READ CHECK
        MEM.NOP=02                      ;NEITHER CPU NOR IB GETS MEM CYCLE
        TEST.WCHK=04                    ;TEST TBUF WITH WRITE CHECK
        WRITE.V.NOCHK=0A                ;WRITE, INHIBIT TRAPS
        WRITE.V.WCHK=0C                 ;WRITE, NORMAL VARIETY
        LOCKWRITE.V.XCHK=0E             ;INTERLOCK WRITE, VIRTUAL ADDRESS
        READ.V.RCHK=10                  ;READ, NORMAL VARIETY
        READ.V.NOCHK=12                 ;READ, INHIBIT TRAPS
        READ.V.MCHK=14                  ;READ FOR MODIFY
        READ.V.IBCHK=16                 ;READ, CHECK CONTHOLLED BY IBUFFER
        READ.V.NEWPC=18                 ;BEGIN NEW INSTRUCTION STREAM
                                        ; DATA GOES TO INSTRUCTION BUFFER
        LOCKREAD.V.NOCHK=1A             ;INTERLOCK READ, INHIBIT CHECK
        LOCKREAD.V.WCHK=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=2E                  ;INTERLOCK WRITE, PHYSICAL
        READ.P=32                       ;READ, PHYSICAL
        READ.INT.SUM=36                 ;INTERRUPT SUMMARY READ
        LOCKREAD.P=3A                   ;INTERLOCK READ, PHYSICAL
        ALLOW.IB.READ=3E                ;GIVE IB A CYCLE IF IT WANTS ONE

MSC/=<29:26>,,DEFAULT=0
        NOP=0                           ;DEFAULT
        CHK.CHM=01                      ;CREATE NEW PSL FOR CHM
        CHK.FLT.OPR=02                  ;UTRAP IF ALU<15>=1, ALU<14:7>=0
        CHK.ODD.ADDR=03
        IRD=04                          ;THIS STATE IS INSTRUCTION DECODE
        LOAD.STATE=05
        LOAD.ACC.CC=06                  ;TAKE CONDITION CODES FROM ACCELERATOR
        READ.RLOG=07                    ;(AND POP RLOG STACK)
        CLR.FPD=08                      ;CLEAR PSL<FPD> BIT
        SET.FPD=09                      ;SET SAME
        CLR.NEST.ERR=0A                 ;CLR NESTED ERROR FLAG IN CPU STATUS
        SET.NEST.ERR=0B                 ;SET SAME
        SECOND.REF=0C                   ;OF UNALIGNED DATA REFERENCE
        RETRY.NO.TRAP=0D                ;APPLY SAVED CONTEXT, INHIBIT TRAPS
        RETRY.TRAP=0E                   ;APPLY SAVED CONTEXT TO THIS REF
        INH.CM.ADDR=0F                  ;ALLOW USE OF FULL 32-BIT ADDRESS
```

```
.TOC    "       Machine definition    : PCK, QK, RAMX, RBMX"

PCK/=<34:32>,.DEFAULT=0          ;ADDRESS COUNT CONTROL
        NOP=0                    ;DEFAULT
        PC_VA=1
        PC_IBA=2
        VA+4=3                   ;VA_VA+4
        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 INSTR BUFFER

QK/=<54:51>,.DEFAULT=0
        NOP=0                    ;DEFAULT, HOLD
        LEFT2=1                  ;DOUBLE SHIFT LEFT 2
        RIGHT2=2                 ;DOUBLE SHIFT RIGHT 2
        LEFT=5
        RIGHT=6
        SHF=8                    ;LOAD SHF, INTEGER FORMAT
        SHF.FL=9                 ;LOAD SHF, UNPACKED FLOATING FORMAT
        DEC.CON=0A               ;DECIMAL CONSTANT = 6'S IN EACH NIBBLE
                                 ;FOR WHICH ALU CRY OUT IS FALSE
        ACCEL=0B                 ;LOAD ACCELERATOR DATA FROM DF BUS
        D=0C
        ID=0E                    ;LOAD ID BUS
        CLR=0F                   ;LOAD ZERO

RAMX/=<77:77>,.DEFAULT=0         ;DATA PATH MIXER TO AMX
        D=0                      ;DEFAULT
        Q=1

RBMX/=<77:77>                    ;DATA PATH MIXER TO BMX.  SAME BIT AS RAMX
        Q=0
        D=1
```

```
.TOC    "       Machine definition    : SCK, SGN, SHF, SI, SMX"

SCK/=<23:23>,.DEFAULT=0         ;SC REGISTER CONTROL
        NOP=0                   ;DEFAULT, HOLD
        LOAD=1                  ;LOAD SMX<09:00>

SGN/=<50:48>,.DEFAULT=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            ;SD_ALU<15>, SS_SS.XOR.ALU<15>
        ADD.SUB=6               ;SD_ALU<15>, SS_SS.XOR.ALU<15>.XOR.IR<1>
        CLR.SD+SS=7             ;CLEAR BOTH

SHF/=<87:85>,.DEFAULT=0         ;ALU SHIFTER CONTROLS
        ALU=0                   ;DEFAULT, SHF_ALU
        LEFT=1                  ;SHF_ALU(L1), INSERT SI CNTL
        RIGHT=2                 ;SHF_ALU(R1), INSERT SI CNTL
        ALU.DT=3                ;SHF_ALU(DT: L0,L1,L2,L3), INSERT 0
        RIGHT2=4                ;SHF_ALU(R2), INSERT SI CNTL
        LEFT3=5                 ;SHF_ALU(L3)

SI/=<57:55>,.DEFAULT=3          ;SHIFT INPUT CONTROLS
                                ;       SHF     D       Q
                                ;       ---     -       -
        DIVD=0                  ;       PSL<N>  Q31     ALU C31
        ASHR=1                  ;       ALU 31  Q0      Q31
        ASHL=2                  ;       0       0       D31
        ZERO=3                  ;       0       0       0
;       SPARE=4
        DIV=5                   ;       Q31     Q31     ALU C31
        MUL+=6                  ;       0       ALU 0,1 0
        MUL-=7                  ;       1       ALU 0,1 1

SMX/=<17:16>                    ;MIXER TO SC
        EALU=0                  ;EALU <9:0>
        FE=1                    ;FE<9:0>
        ALU=2                   ;ALU<09:00>
        ALU.EXP=3               ;ALU<14:07>
```

```
.TOC    "       Machine definition      : SPO, SPO.AC, SPO.ACN, SPO.ACN11, SPO.R"

SPO/=<41:35>,.DEFAULT=0                 ;SCRATCH PAD UPCODE, 7 BITS
        NOP=0                           ;DEFAULT
        LOAD.LC.SC=6                    ;LOAD LC, ADR=SC[03:00]
        WRITE.RC.SC=7                   ;WRITE RC, ADR=SC[03:00]

SPO.AC/=<41:38>                         ;4 FUNCTION BITS OF SPO FIELD
        LOAD.LAB=1                      ;LOAD LA, LB FROM R(ACN)
        LOAD.LA=2                       ;LOAD LA_RN, HOLD LB
        WRITE.RAB=3                     ;WRITE RA, RB (ACN)

SPO.ACN/=<37:35>                ;AC NUMBER IN SPO 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 P+1         SP1 R+1

SPO.ACN11/=<37:35>              ;AC NUMBER IN SPO FIELD -- 11 MODE
                                ;-11 MODE         RA              RB
        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>

SPO.R/=<41:39>                  ;SCRATCH PAD FUNCS WITH LOW 4 BITS OF SP AS ADR
        LOAD.LC=2               ;LOAD LC, ADR=SPO.RN
        WRITE.RC=3              ;WRITE RC
        LOAD.LAB=4              ;LOAD LA, LB
        WRITE.RAB=5             ;WRITE RA, RB
        LOAD.LAB1.WRITE.RC=6    ;LOAD LA, LB[R1], AND WRITE RC[RN]
        LOAD.LC.WRITE.RAB1=7    ;LOAD LC[RN], AND WRITE RA, RB[R1]
```

```
.TOC    "       Machine definition    : SPO.RAB, SPO.RC, SUB, VAK"

SPO.RAB/=<38:35>                ;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                   ;R14 = STACK POINTER
        R15=0F                  ;R15 = PC, TO SOFTWARE, SCRATCH TO UCODE

SPO.RC/=<38:35>                 ;RC LOCATIONS
        T0=0
        T1=1
        T2=2
        T3=3
        T4=4
        T5=5
        T6=6
        T7=7
        LC.SV=8                 ;MEM MGMT SAVES LC HERE
        VA.SV=9
        PTE.VA=0A
        PTE.PA=0B
        PC.SV=0C
        SC.SV=0D
        VA.REF=0E
        MBIT.VA=0F
        PTE.MASK=0F

SUB/=<65:64>,.DEFAULT=0         ;SUBROUTINE CONTROL
        NOP=0                   ;DEFAULT
        CALL=1                  ;PUSH UPC OF THIS MICROINSTRUCTION
                                ; ONTO USTACK
        RET=2                   ;"OR" TOP OF USTACK TO UPC
                                ; AND POP USTACK
        SPEC=3                  ;REPLACE LOW 8 BITS OF NEXT
                                ; UPC WITH SPECIFIER DECODE FROM
                                ; INSTRUCTION BUFFER

VAK/=<25:25>,.DEFAULT=0
        NOP=0                   ;DEFAULT
        LOAD=1                  ;LOAD VA
```

```
.TOC    "       Machine definition      : Validity checks"

        .SET/V0=<.NOT[<NATIVE>]>
        .SET/V1=<NATIVE>



.TOC    "       Macro definition        : Regions"

                    ;  +--------+-----+-----------------------+
                    ;  |      0 | 0k  |                       |
                    ;  |        |     |                       |
                    ;  |        |     |                       |
                    ;  |        |     |                       |
                    ;  |        | to  | PCS                   |
                    ;  |        |     |                       |
                    ;  |        |     |                       |
                    ;  |        |     |                       |
                    ;  |   4095 | 4k  |                       |
                    ;  +--------+-----+-----------------------+
.SET/WCSR1L=1000    ;  |   4096 | 4k  |                       |
.SET/WCSR1H=113F    ;  |        |     |                       | Note : 1140 to 11FF is
                    ;  |        | to  | DEC'S WCS region      |        the FPLA trap
.SET/WCSR2L=1200    ;  |        |     |                       |        address region
.SET/WCSR2H=17FF    ;  |   6143 | 6k  |                       |
                    ;  +--------+-----+-----------------------+
.SET/WCSR3L=1800    ;  |   6144 | 6k  |                       |
                    ;  |        |     |                       |
                    ;  |        | to  | User or G&H WCS       |
                    ;  |        |     |                       |
.SET/WCSR3H=1BFF    ;  |   7167 | 7k  |                       |
                    ;  +--------+-----+-----------------------+
;           1C00    ;  |   7168 | 7k  |                       |
                    ;  |        |     |                       |
                    ;  |        | to  | User WCS              |
                    ;  |        |     |                       |
                    ;  |   8191 | 8k  |                       |
;           1FFF    ;  +--------+-----+-----------------------+
```

```
.TOC    "       Macro definition      : Register transfer macros"

ALU_-1                          "AMX/RAMX.OXT,DT/LONG,ALU/NOTA"
ALU_0(A)                        "AMX/RAMX.OXT,DT/LONG,ALU/A"
ALU_0+D                         "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A+B"
ALU_0+D+1                       "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A+B+1"
ALU_0+K[]                       "KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/A+B"
ALU_0+K[]+1                     "KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/A+B+1"
ALU_0+LB+1                      "AMX/RAMX.OXT,DT/LONG,BMX/LB,ALU/A+B+1"
ALU_0+LC                        "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A+B"
ALU_0+LC+1                      "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A+B+1"
ALU_0+MASK+1                    "AMX/RAMX.OXT,DT/LONG,BMX/MASK,ALU/A+B+1"
ALU_0+Q                         "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B"
ALU_0+Q+1                       "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B+1"
ALU_0-D                         "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B"
ALU_0-D-1                       "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B-1"
ALU_0-K[]                       "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B"
ALU_0-K[]-1                     "KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/A-B-1"
ALU_0-LB                        "AMX/RAMX.OXT,DT/LONG,BMX/LB,ALU/A-B"
ALU_0-LC                        "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A-B"
ALU_0-LC-1                      "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A-B-1"
ALU_0-Q                         "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B"
ALU_0-Q-1                       "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B-1"
ALU_0[]D                        "ALU/@1,AMX/RAMX.OXT,LONG,BMX/RBMX,RBMX/D"
ALU_0[]LC                       "ALU/@1,AMX/RAMX.OXT,LONG,BMX/LC"
ALU_D                           "RAMX/D,AMX/RAMX,ALU/A"
ALU_D(B)                        "RBMX/D,BMX/RBMX,ALU/B"
ALU_D+K[]                       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B"
ALU_D+K[]+1                     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1"
ALU_D+K[].RLOG                  "AMX/RAMX,RAMX/D,KMX/@1,BMX/KMX,ALU/A+B.RLOG"
ALU_D+LB                        "RAMX/D,AMX/RAMX,BMX/LB,ALU/A+B"
ALU_D+LC                        "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B"
ALU_D+LC+1                      "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+1"
ALU_D+LC+PSL.C                  "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+PSL.C"
ALU_D+Q                         "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B"
ALU_D+Q+1                       "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1"
ALU_D+Q+PSL.C                   "ALU/A+B+PSL.C,AMX/RAMX,BMX/RBMX,RBMX/Q,RAMX/D"
ALU_D+RLOG                      "ALU/A+B,AMX/RAMX,RAMX/D,BMX/0,PSC/READ.RLOG"
ALU_D-K[]                       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B"
ALU_D-K[]-1                     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1"
ALU_D-LB                        "RAMX/D,AMX/RAMX,BMX/LB,ALU/A-B"
ALU_D-LB.RLOG                   "RAMX/D,AMX/RAMX,BMX/LB,ALU/A-B.RLOG"
ALU_D-LC                        "RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B"
ALU_D-LC-1                      "RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B-1"
ALU_D-Q                         "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B"
ALU_D-Q-1                       "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B-1"
ALU_D.0XT[]                     "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A"
ALU_D.0XT[]+K[]                 "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_D.0XT[]+LC                  "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/LC"
ALU_D.0XT[]+Q                   "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/RBMX,RBMX/Q"
ALU_D.0XT[]-K[]                 "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B"
```

```
ALU_D.OXT[]-Q                "RAMX/D,AMX/RAMX.OXT,DT/@1,RBMX/Q,BMX/RBMX,ALU/A-B"
ALU_D.OXT[].AND.K[]          "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/AND"
ALU_D.OXT[].ANDNOT.K[]       "ALU/ANDNOT,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/KMX,KMX/@2"
ALU_D.OXT[].OR.Q             "RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/RBMX,ALU/OR"
ALU_D.AND.K[]                "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND"
ALU_D.AND.MASK               "RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND"
ALU_D.ANDNOT.K[]             "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_D.ANDNOT.MASK            "RAMX/D,AMX/RAMX,BMX/MASK,ALU/ANDNOT"
ALU_D.ANDNOT.Q               "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT"
ALU_D.OR.K[]                 "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR"
ALU_D.OR.LC                  "RAMX/D,AMX/RAMX,BMX/LC,ALU/OR"
ALU_D.OR.Q                   "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR"
ALU_D.OR.RC[]                "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR"
ALU_D.ORNOT.MASK             "RAMX/D,AMX/RAMX,BMX/MASK,ALU/ORNOT"
ALU_D.SXT[]                  "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A"
ALU_D.SXT[]+K[]              "RAMX/D,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_D.SXT[]+Q                "RAMX/D,AMX/RAMX.SXT,DT/@1,BMX/RBMX,ALU/A+B"
ALU_D.SXT[].ANDNOT.K[]       "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/ANDNOT,BMX/KMX,KMX/@2"
ALU_D.SXT[].AND.K[]          "RAMX/D,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/AND"
ALU_D.XOR.K[]                "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR"
ALU_D.XOR.LC                 "RAMX/D,AMX/RAMX,BMX/LC,ALU/XOR"
ALU_D.XOR.Q                  "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/XOR"
ALU_D.XOR.RC[]               "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR"
ALU_D.XOR.R[]                "RAMX/D,AMX/RAMX,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/LB,ALU/XOR"
ALU_D[]K[]                   "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/@1"
ALU_D[]LB                    "ALU/@1,AMX/RAMX,RAMX/D,BMX/LB"
ALU_D[]LC                    "RAMX/D,AMX/RAMX,BMX/LC,ALU/@1"
ALU_D[]Q                     "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1"
ALU_K[]                      "KMX/@1,BMX/KMX,ALU/B"
ALU_LA                       "AMX/LA,ALU/A"
ALU_LA+K[]                   "AMX/LA,KMX/@1,BMX/KMX,ALU/A+B"
ALU_LA+K[]+1                 "ALU/A+B+1,AMX/LA,BMX/KMX,KMX/@1"
ALU_LA+K[].RLOG              "AMX/LA,KMX/@1,BMX/KMX,ALU/A+B.RLOG"
ALU_LA+LB                    "AMX/LA,BMX/LB,ALU/A+B"
ALU_LA+LC                    "ALU/A+B,AMX/LA,BMX/LC"
ALU_LA+LC+1                  "ALU/A+B+1,AMX/LA,BMX/LC"
ALU_LA+LC+PSL.C              "ALU/A+B+PSL.C,AMX/LA,BMX/LC"
ALU_LA+Q                     "ALU/A+B,AMX/LA,BMX/RBMX,RBMX/Q"
ALU_LA-D                     "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B"
ALU_LA-D-1                   "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B-1"
ALU_LA-K[]                   "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B"
ALU_LA-K[]-1                 "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B-1"
ALU_LA-K[].RLOG              "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B.RLOG"
ALU_LA-LC                    "ALU/A-B,AMX/LA,BMX/LC"
ALU_LA-Q                     "ALU/A-B,AMX/LA,BMX/RBMX,RBMX/Q"
ALU_LA-Q-1                   "ALU/A-B-1,AMX/LA,BMX/RBMX,RBMX/Q"
ALU_LA.AND.K[]               "AMX/LA,KMX/@1,BMX/KMX,ALU/AND"
ALU_LA.AND.LC                "ALU/AND,AMX/LA,BMX/LC"
ALU_LA.ANDNOT.K[]            "AMX/LA,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_LA.ANDNOT.MASK           "AMX/LA,BMX/MASK,ALU/ANDNOT"
```

```
ALU_LA.OR.K[]                "ALU/OR,AMX/LA,BMX/KMX,KMX/@1"
ALU_LA.XOR.LC                "AMX/LA,BMX/LC,ALU/XOR"
ALU_LA[]D                    "AMX/LA,RBMX/D,BMX/RBMX,ALU/@1"
ALU_LA[]LB                   "AMX/LA,BMX/LB,ALU/@1"
ALU_LA[]Q                    "AMX/LA,RBMX/Q,BMX/RBMX,ALU/@1"
ALU_LB                       "BMX/LB,ALU/B"
ALU_LC                       "BMX/LC,ALU/B"
ALU_NOT.D                    "ALU/NOTA,AMX/RAMX,RAMX/D"
ALU_NOT.K[]                  "BMX/KMX,KMX/@1,ALU/ORNOT,AMX/RAMX.0XT,DT/LONG"
ALU_NOT.RC[]                 "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,AMX/RAMX.0XT,DT/LONG,ALU/ORNOT"
ALU_PACK.FP                  "BMX/PACKED.FL,ALU/B"
ALU_PC                       "BMX/PC,ALU/B"
ALU_Q                        "RAMX/Q,AMX/RAMX,ALU/A"
ALU_Q(B)                     "RBMX/Q,BMX/RBMX,ALU/B"
ALU_Q+K[]                    "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B"
ALU_Q+K[]+1                  "ALU/A+B+1,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@1"
ALU_Q+LB                     "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B"
ALU_Q+LB+1                   "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B+1"
ALU_Q+LC                     "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B"
ALU_Q+LC+1                   "ALU/A+B+1,AMX/RAMX,RAMX/Q,BMX/LC"
ALU_Q+LC+PSL.C               "ALU/A+B+PSL.C,AMX/RAMX,RAMX/Q,BMX/LC"
ALU_Q+MASK                   "ALU/A+B,AMX/RAMX,RAMX/Q,BMX/MASK"
ALU_Q-D                      "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B"
ALU_Q-D-1                    "ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D"
ALU_Q-K[]                    "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B"
ALU_Q-LB                     "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A-B"
ALU_Q-LC                     "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B"
ALU_Q-MASK-1                 "ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/MASK"
ALU_Q.0XT[]                  "RAMX/Q,AMX/RAMX.0XT,DT/@1,ALU/A"
ALU_Q.0XT[]+D                "ALU/A+B,AMX/RAMX.0XT,DT/@1,BMX/RBMX,RBMX/D,RAMX/Q"
ALU_Q.0XT[]+D+1              "ALU/A+B+1,AMX/RAMX.0XT,DT/@1,BMX/RBMX,RAMX/Q,RBMX/D"
ALU_Q.0XT[]+K[]              "ALU/A+B,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.0XT[]-D                "ALU/A-B,RAMX/Q,AMX/RAMX.0XT,DT/@1,BMX/RBMX"
ALU_Q.0XT[]-K[]              "ALU/A-B,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.0XT[].ANDNOT.K[]       "ALU/ANDNOT,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.0XT[].OR.K[]           "ALU/OR,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.0XT[].OR.D             "ALU/OR,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/RBMX,RBMX/D"
ALU_Q.AND.D                  "AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D,ALU/AND"
ALU_Q.AND.K[]                "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND"
ALU_Q.ANDNOT.K[]             "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_Q.ANDNOT.MASK            "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ANDNOT"
ALU_Q.ANDNOT.R[]             "ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/LB,SPO.R/LOAD.LAB,SPO.RAB/@1"
ALU_Q.OR.K[]                 "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR"
ALU_Q.OR.LC                  "RAMX/Q,AMX/RAMX,BMX/LC,ALU/OR"
ALU_Q.ORNOT.K[]              "ALU/ORNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@1"
ALU_Q.SXT[]                  "ALU/A,AMX/RAMX.SXT,DT/@1,RAMX/Q"
ALU_Q.SXT[]+K[]              "RAMX/Q,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_Q.SXT[]+LB               "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/LB,ALU/A+B"
ALU_Q.SXT[]+LB+1             "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/LB,ALU/A+B+1"
ALU_Q.SXT[]+PC               "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/PC,ALU/A+B"
```

```
ALU_Q.SXT[].ANDNOT.K[]          "ALU/ANDNOT,AMX/RAMX.SXT,RAMX/Q,BMX/KMX,KMX/@2,DT/@1"
ALU_Q.XOR.D                     "RAMX/Q,AMX/RAMX,BMX/RBMX,RBMX/D,ALU/XOR"
ALU_Q.XOR.K[]                   "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR"
ALU_Q.XOR.LC                    "RAMX/Q,AMX/RAMX,BMX/LC,ALU/XOR"
ALU_Q.XOR.RC[]                  "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR"
ALU_Q[]D                        "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/@1"
ALU_R(DST)                      "SPO.AC/LOAD.LAB,SPO.ACN11/DST.DST,AMX/LA,ALU/A"
ALU_R(SC).ANDNOT.K[]            "SPO.AC/LOAD.LAB,SPO.ACN/SC,AMX/LA,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_R(SP1)+K[].RLOG             "SPO.AC/LOAD.LAB,SPO.ACN/SP1.SP1,AMX/LA,KMX/@1,BMX/KMX,ALU/A+B.RLOG"
ALU_RC(SC)                      "SPO/LOAD.LC.SC,BMX/LC,ALU/B"
ALU_RC[]                        "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B"
ALU_RLOG                        "BMX/0,ALU/B,MSC/READ.RLOG"
ALU_R[]                         "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A"
ALU_R[]-K[]                     "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/A-B"
ALU_R[].AND.K[]                 "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND"
ALU_R[].AND.LC                  "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/LC,ALU/AND"
ALU_R[].ANDNOT.K[]              "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/ANDNOT"
ALU_R[].ANDNOT.MASK             "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/MASK,ALU/ANDNOT"
ALU_R[].OR.K[]                  "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/OR"
ALU_R[].ORNOT.K[]               "ALU/ORNOT,AMX/LA,BMX/KMX,SPO.R/LOAD.LAB,SPO.RAB/@1,KMX/@2"
ALU_R[].XOR.K[]                 "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/XOR"

CACHE.P_D[]                     "VAK/NOP,MCT/WRITE.P,DT/@1,DK/NOP"
CACHE[]_D                       "VAK/NOP,MCT/WRITE.V.WCHK,MSC/@1,DK/NOP"
CACHE_D(QUAD)                   "MCT/EXTWRITE.P,LONG,VAK/NOP,DK/NOP"
CACHE_D.INST.DEP                "VAK/NOP,MCT/WRITE.V.WCHK,DT/INST.DEP,DK/NOP"
CACHE_D[]                       "VAK/NOP,MCT/WRITE.V.WCHK,DT/@1,DK/NOP"
CACHE_D[].LK                    "VAK/NOP,MCT/LOCKWRITE.V.XCHK,DT/@1,DK/NOP"
CACHE_D[].NOCHK                 "VAK/NOP,MCT/WRITE.V.NOCHK,DT/@1,DK/NOP"

D&Q_D+Q                         "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF,QK/SHF"
D&RC[]_PC                       "BMX/PC,ALU/B,SHF/ALU,DK/SHF,SPO.R/WRITE.RC,SPO.RC/@1"
D&VA_ALU                        "VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_D+LC                       "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_D+Q                        "D_D+Q,VAK/LOAD"
D&VA_D-K[]                      "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_LA                         "AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_LB                         "BMX/LB,ALU/B,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_Q                          "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,DK/Q"
D&VA_Q+LB.PC                    "RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,DK/SHF"

D[]_CACHE                       "VAK/NOP,MCT/READ.V.RCHK,DT/@1,DK/NOP"
D[]_CACHE.IBCHK                 "VAK/NOP,MCT/READ.V.IBCHK,DT/@1,DK/NOP"
D[]_CACHE.LK                    "VAK/NOP,MCT/LOCKREAD.V.WCHK,DT/@1,DK/NOP"
D[]_CACHE.NOCHK                 "VAK/NOP,MCT/READ.V.NOCHK,DT/@1,DK/NOP"
D[]_CACHE.P                     "VAK/NOP,MCT/READ.P,DT/@1,DK/NOP"
D[]_CACHE.WCHK                  "VAK/NOP,MCT/READ.V.WCHK,DT/@1,DK/NOP"
```

```
D_0                     "DK/CLR"
D_0+K[]+1               "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_0+LC+1                "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,DK/SHF"
D_0-D                   "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_0-K[]                 "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
D_0-Q                   "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_0-Q-1                 "ALU_0-Q-1,D_ALU"
D_ACCEL&SYNC            "DK/ACCEL,ACF/SYNC"
D_ALU                   "SHF/ALU,DK/SHF"
D_ALU(FRAC)             "SHF/ALU,DK/SHF.FL"
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/RIGH12,DK/SHF"
D_BLANK                 "D_K[,20]"
D_CACHE.INST.DEP        "VAK/NOP,MCT/READ,V.IBCHK,DT/INST.DEP,DK/NOP"
D_CACHE.LK[]            "VAK/NOP,MCT/LOCKREAD.V.WCHK,MSC/@1,DK/NOP"
D_CACHE.WCHK[]          "VAK/NOP,MCT/READ.V.WCHK,MSC/@1,DK/NOP"
D_CACHE[]               "VAK/NOP,MCT/READ.V.RCHK,MSC/@1,DK/NOP"
D_D(FRAC)               "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,DK/SHF.FL"
D_D+K[]                 "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF"
D_D+K[]+1               "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_D+LB                  "RAMX/D,AMX/RAMX,BMX/LB,ALU/A+B,SHF/ALU,DK/SHF"
D_D+LC                  "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,DK/SHF"
D_D+LC+PSL.C            "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+PSL.C,SHF/ALU,DK/SHF"
D_D+Q                   "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF"
D_D+Q+1                 "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_D-K[]                 "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
D_D-LC                  "RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,DK/SHF"
D_D-Q                   "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_D-Q-1                 "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B-1,SHF/ALU,DK/SHF"
D_D.OXT[]               "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_D.OXT[]+K[]           "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF"
D_D.OXT[]+Q             "ALU/A+B,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RBMX/Q,D_ALU"
D_D.OXT[]+Q+1           "RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/RBMX,ALU/A+B+1,D_ALU"
D_D.OXT[].ANDNOT.K[]    "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.OXT[].OR.Q          "RAMX/D,AMX/RAMX.OXT,DT/@1,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,DK/SHF"
D_D.OXT[].XOR.Q         "DK/SHF,ALU/XOR,SHF/ALU,AMX/RAMX.OXT,RAMX/D,DT/@1,RBMX/Q,BMX/RBMX"
D_D.OXT[].XOR.RC[]      "RAMX/D,AMX/RAMX.OXT,DT/@1,SPO.R/LOAD.LC,SPO.RC/@2,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF"
D_D.AND.K[]             "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_D.AND.K[].LEFT2       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU.DT,DT/LONG,DK/SHF"
D_D.AND.K[].RIGHT       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,DK/SHF"
D_D.AND.LC              "RAMX/D,AMX/RAMX,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D_D.AND.MASK            "RAMX/D,AMX/RAMX.BMX/MASK.ALU/AND.SHF/ALU.DK/SHF"
D_D.AND.Q               "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/AND,SHF/ALU,DK/SHF"
D_D.AND.RC[]            "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D_D.ANDNOT.K[]          "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.ANDNOT.LC           "RAMX/D,AMX/RAMX,BMX/LC,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.ANDNOT.PSWZ         "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.4,SHF/ALU"
D_D.ANDNOT.Q            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
```

```
D_D.ANDNOT.RC[]          "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.LEFT                 "DK/LEFT"
D_D.LEFT2                "DK/LEFT2"
D_D.OR.ASCII             "D_D.OR.K[.30]"
D_D.OR.K[]               "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF"
D_D.OR.PSWC              "DK/SHF,ALU/OR,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.1,SHF/ALU"
D_D.OR.PSWV              "DK/SHF,ALU/OR,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.2,SHF/ALU"
D_D.OR.Q                 "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,DK/SHF"
D_D.OR.RC[]              "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,DK/SHF"
D_D.OR.R[]               "ALU/OR,AMX/RAMX,RAMX/D,BMX/LB,SPO.R/LOAD.LAB,SPO.RAB/@1,DK/SHF"
D_D.ORNOT.MASK           "RAMX/D,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,DK/SHF"
D_D.RIGHT                "DK/RIGHT"
D_D.RIGHT(B)             "RBMX/D,BMX/RBMX,ALU/B,SHF/RIGHT,DK/SHF"
D_D.RIGHT2               "DK/RIGHT2"
D_D.SWAP                 "DK/BYTE.SWAP"
D_D.SXT[]                "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_D.SXT[].RIGHT          "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/RIGHT,DK/SHF"
D_D.XOR.K[]              "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR,SHF/ALU,DK/SHF"
D_D.XOR.LC               "RAMX/D,AMX/RAMX,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF"
D_D.XOR.Q                "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/XOR,SHF/ALU,DK/SHF"
D_DAL.NORM               "DK/DAL.SV"
D_DAL.SC                 "DK/DAL.SC"
D_D[]K[]                 "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/@1,SHF/ALU,DK/SHF"
D_D[]MASK                "RAMX/D,AMX/RAMX,BMX/MASK,ALU/@1,SHF/ALU,DK/SHF"
D_D[]Q                   "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1,SHF/ALU,DK/SHF"
D_INT.SUM                "MCT/READ.INT.SUM,DK/NOP"
D_K[]                    "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,DK/SHF"
D_K[].RIGHT              "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT,DK/SHF"
D_K[].RIGHT2             "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT2,DK/SHF"
D_LA                     "AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_LA(FRAC)               "AMX/LA,ALU/A,SHF/ALU,DK/SHF.FL"
D_LA+D+PSL.C             "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B+PSL.C,SHF/ALU,DK/SHF"
D_LA-D                   "DK/SHF,ALU/A-B,AMX/LA,BMX/RBMX,RBMX/D,SHF/ALU"
D_LA-K[]                 "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
D_LA.AND.K[]             "AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_LA.RIGHT               "AMX/LA,ALU/A,SHF/RIGHT,DK/SHF"
D_LB                     "BMX/LB,ALU/B,SHF/ALU,DK/SHF"
D_LB.PC                  "BMX/PC.OR.LB,ALU/B,SHF/ALU,DK/SHF"
D_LC                     "BMX/LC,ALU/B,SHF/ALU,DK/SHF"
D_LC(FRAC)               "BMX/LC,ALU/B,SHF/ALU,DK/SHF.FL"
D_NOT.D                  "RAMX/D,AMX/RAMX,ALU/NOTA,SHF/ALU,DK/SHF"
D_NOT.K[]                "KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT,SHF/ALU,DK/SHF"
D_NOT.MASK               "BMX/MASK,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT,SHF/ALU,DK/SHF"
D_NOT.Q                  "RAMX/Q,AMX/RAMX,ALU/NOTA,SHF/ALU,DK/SHF"
D_NOT.R[]                "LA_RA[@1],AMX/LA,ALU/NOTA,D_ALU"
D_PACK.FP                "BMX/PACKED.FL,ALU/B,SHF/ALU,DK/SHF"
D_PACK.FP.LEFT           "BMX/PACKED.FL,ALU/B,SHF/LEFT,DK/SHF"
D_PC                     "BMX/PC,ALU/B,SHF/ALU,DK/SHF"
D_PC.LEFT                "BMX/PC,ALU/B,SHF/LEFT,DK/SHF"
D_Q                      "DK/Q"
```

```
D_Q(FRAC)           "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,DK/SHF.FL"
D_Q+D               "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF"
D_Q+K[]             "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF"
D_Q+LB              "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B,SHF/ALU,DK/SHF"
D_Q+PC              "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,DK/SHF"
D_Q-D               "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_Q-D-1             "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B-1,SHF/ALU,DK/SHF"
D_Q-K[]             "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
D_Q-K[]-1           "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1,SHF/ALU,DK/SHF"
D_Q-PCSV            "RAMX/Q,AMX/RAMX,BMX/0,MSC/READ.RLOG,ALU/A-B,SHF/ALU,DK/SHF"
D_Q.OXT[]           "RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_Q.AND.K[]         "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_Q.AND.LC          "RAMX/Q,AMX/RAMX,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D_Q.AND.MASK        "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,DK/SHF"
D_Q.AND.RC[]        "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D_Q.ANDNOT.D        "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_Q.ANDNOT.K[]      "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_Q.ANDNOT.MASK     "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_Q.ANDNOT.PSWC     "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.1,SHF/ALU"
D_Q.ANDNOT.PSWN     "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.8,SHF/ALU"
D_Q.ANDNOT.PSWZ     "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.4,SHF/ALU"
D_Q.LEFT            "RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT,DK/SHF"
D_Q.OR.K[]          "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF"
D_Q.OR.PSWC         "DK/SHF,ALU/OR,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.1,SHF/ALU"
D_Q.OR.RC[]         "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,DK/SHF"
D_Q.ORNOT.MASK      "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,DK/SHF"
D_Q.RIGHT           "RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT,DK/SHF"
D_Q.RIGHT2          "RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT2,DK/SHF"
D_Q.SXT[]           "RAMX/Q,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_Q.XOR.RC[]        "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF"
D_Q[]D              "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/@1,SHF/ALU,DK/SHF"
D_Q[]K[]            "ALU/@1,SHF/ALU,DK/SHF,BMX/KMX,KMX/@2,AMX/RAMX,RAMX/Q"
D_Q[]MASK           "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/@1,SHF/ALU,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"
D_RC(SC)            "SPO/LOAD.LC.SC,BMX/LC,ALU/B,SHF/ALU,DK/SHF"
D_RC[]              "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SHF/ALU,DK/SHF"
D_RLOG              "BMX/0,MSC/READ.RLOG,ALU/B,SHF/ALU,DK/SHF"
D_RLOG.RIGHT        "BMX/0,MSC/READ.RLOG,ALU/B,SHF/RIGHT,DK/SHF"
D_R[]               "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_R[](FRAC)         "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF.FL"
D_R[].AND.K[]       "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_R[].OR.K[]        "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF"
D_R[].ORNOT.K[]     "LAB_R[@1],AMX/LA,BMX/KMX,KMX/@2,ALU/ORNOT,D_ALU"

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+FE                  "EBMX/FE,EALU/A+B"
EALU_SC+K[]                 "KMX/@1,EBMX/KMX,EALU/A+B"
EALU_SC-FE                  "EBMX/FE,EALU/A-B"
EALU_SC-K[]                 "KMX/@1,EBMX/KMX,EALU/A-B"
EALU_SC.ANDNOT.K[]          "KMX/@1,EBMX/KMX,EALU/ANDNOT"
EALU_STATE                  "EALU/A,MSC/LOAD.STATE"

FE&SC_K[]                   "KMX/@1,EBMX/KMX,EALU/B,FEK/LOAD,SMX/EALU,SCK/LOAD"
FE_0(A)                     "AMX/RAMX.0XT,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_NABS(SC-FE)              "EALU/NABS.A-B,EBMX/FE,FEK/LOAD"
FE_NABS(SC-LA(EXP))         "AMX/LA,EBMX/AMX.EXP,EALU/NABS.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+1                     "EALU/A+1,FEK/LOAD"
FE_SC+FE                    "EBMX/FE,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-FE                    "EBMX/FE,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-SHF.VAL               "EBMX/SHF.VAL,EALU/A-B,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.OR.K[]                "EALU/OR,EBMX/KMX,KMX/@1,FEK/LOAD"
FE_SHF.VAL                  "EBMX/SHF.VAL,EALU/B,FEK/LOAD"
FE_STATE                    "MSC/LOAD.STATE,EALU/A,FEK/LOAD"

ID(SC)_D                    "CID/WRITE.SC"
ID[]_D                      "CID/WRITE.KMX,ID.ADDR/@1"
ID_D&NO.SYNC                "CID/WRITE.KMX,ADS/IBA,KMX/SP1.CON"
ID_D.SYNC                   "CID/WRITE.KMX,ADS/IBA,KMX/SP1.CON,ACF/SYNC"

K[]                         "KMX/@1"

LAB_R(DST)                  "SPO.AC/LOAD.LAB,SPO.ACN11/DST.DST"
LAB_R(PRN)                  "SPO.AC/LOAD.LAB,SPO.ACN/PRN"
LAB_R(PRN+1)                "SPO.AC/LOAD.LAB,SPO.ACN/PRN+1"
LAB_R(SC)                   "SPO.AC/LOAD.LAB,SPO.ACN/SC"
LAB_R(SP1)                  "SPO.AC/LOAD.LAB,SPO.ACN/SP1.SP1"
LAB_R(SP1+1)                "SPO.AC/LOAD.LAB,SPO.ACN/SP1+1"
LAB_R1&RC[]_0               "ALU_0(A),LAB_R1&RC[@1]_ALU"
LAB_R1&RC[]_0+LC+1          "ALU/A+B+1,AMX/RAMX.0XT,DT/LONG,BMX/LC,SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/@1,SHF/ALU"
LAB_R1&RC[]_0-D             "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/@1,ALU/A-B,AMX/RAMX.0XT,DT/LONG,BMX/RBMX,RBMX/D,SHF/ALU"
LAB_R1&RC[]_ALU             "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/@1,SHF/ALU"
```

```
LAB_R1&RC[]_ALU.RIGHT2          "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/@1,SHF/RIGHT2"
LAB_R1&RC[]_D+LC                "ALU_D+LC,LAB_R1&RC[@1]_ALU"
LAB_R1&RC[]_D.OXT[]+K[]         "ALU_D.OXT[@2]+K[@3],LAB_R1&RC[@1]_ALU"
LAB_R1&RC[]_Q-K[]               "ALU_Q-K[@2],LAB_R1&RC[@1]_ALU"
LAB_R[]                         "SPO.R/LOAD.LAB,SPO.RAB/@1"

LA_R(DST)&LB_R(SRC)             "SPO.AC/LOAD.LAB,SPO.ACN11/DST.SRC"
LA_R(SP2)&LB_R(SP1)             "SPO.AC/LOAD.LAB,SPO.ACN/SP2.SP1"
LA_RA[]                         "SPO.AC/LOAD.LA,SPO.RAB/@1"
LC_RC(SC)                       "SPO/LOAD.LC,SC"
LC_RC[]                         "SPO.R/LOAD.LC,SPO.RC/@1"
LC_RC[]&R1_(LA+LB).LEFT         "AMX/LA,BMX/LB,ALU/A+B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
LC_RC[]&R1_(LA+LB+PSL.C).LEFT   "AMX/LA,BMX/LB,ALU/A+B+PSL.C,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
LC_RC[]&R1_(LA+LB.RLOG).LEFT    "AMX/LA,BMX/LB,ALU/A+B.RLOG,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
LC_RC[]&R1_(LA-LB).LEFT         "AMX/LA,BMX/LB,ALU/A-B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
LC_RC[]&R1_(LA-LB.RLOG).LEFT    "AMX/LA,BMX/LB,ALU/A-B.RLOG,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
LC_RC[]&R1_ALU                  "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU"
LC_RC[]&R1_D                    "ALU_D,LC_RC[@1]&R1_ALU"
LC_RC[]&R1_LA+K[]               "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU,ALU/A+B,AMX/LA,BMX/KMX,KMX/@2"
LC_RC[]&R1_LA-K[]               "ALU_LA-K[@2],LC_RC[@1]&R1_ALU"
LC_RC[]&R1_LB                   "ALU_LB,LC_RC[@1]&R1_ALU"
LC_RC[]&R1_Q                    "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU,ALU/A,AMX/RAMX,RAMX/Q"

N&Z_ALU                         "CCK/NZ_ALU.VC_VC"
N&Z_ALU.V&C_0                   "CCK/NZ_ALU.VC_0"
N_AMX.Z_TST                     "CCK/N_AMX.Z_TST.VC_VC"
PC&VA_ALU                       "VAK/LOAD,PCK/PC_VA"
PC&VA_D                         "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
PC&VA_D+K[]                     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_D-K[]                     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
PC&VA_D-PC                      "RAMX/D,AMX/RAMX,BMX/PC,ALU/A-B,VAK/LOAD,PCK/PC_VA"
PC&VA_D.OXT[]                   "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A,VAK/LOAD,PCK/PC_VA"
PC&VA_D.OXT[]+PC                "RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_D.SXT[]+PC                "RAMX/D,AMX/RAMX.SXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_K[]                       "KMX/@1,BMX/KMX,ALU/B,VAK/LOAD,PCK/PC_VA"
PC&VA_PC                        "BMX/PC,ALU/B,VAK/LOAD,PCK/PC_VA"
PC&VA_Q                         "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
PC&VA_Q+PC                      "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_Q-D                       "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
PC&VA_Q-K[]                     "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
PC&VA_Q.SXT[]+PC                "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_RC[]                      "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,VAK/LOAD,PCK/PC_VA"
PC&VA_R[].ANDNOT.K[]            "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/ANDNOT,VAK/LOAD,PCK/PC_VA"

PC_PC+1                         "PCK/PC+1"
PC_PC+2                         "PCK/PC+2"
PC_PC+4                         "PCK/PC+4"
PC_PC+N                         "PCK/PC+N"
PC_Q+PC                         "ALU/A+B,VAK/LOAD,PCK/PC_VA,BMX/PC,AMX/RAMX,RAMX/Q"
PC_VA                           "PCK/PC_VA"
```

```
PC_VIBA                     "PCK/PC_IBA"
PSL<C>_AMX0                 "CCK/C_AMX0"

Q&VA_ALU                    "VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_D                      "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_D+LC                   "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_LA                     "AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_Q+LB.PC                "RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF"

QD_(Q+LB)D.RIGHT2           "ALU_Q+LB,Q_ALU.RIGHT2,D_D.RIGHT2"
QD_(Q+LC)D.RIGHT2           "ALU_Q+LC,Q_ALU.RIGHT2,D_D.RIGHT2"
QD_(Q-LB)D.RIGHT2           "ALU_Q-LB,Q_ALU.RIGHT2,D_D.RIGHT2"
QD_(Q-LC)D.RIGHT2           "ALU_Q-LC,Q_ALU.RIGHT2,D_D.RIGHT2"
QD_QD.RIGHT2                "ALU_Q,Q_ALU.RIGHT2,D_D.RIGHT2"

Q_0                         "QK/CLR"
Q_0+LC+1                    "ALU/A+B+1,AMX/RAMX.0XT,DT/LONG,SHF/ALU,QK/SHF,BMX/LC"
Q_0+MASK+1                  "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,QK/SHF"
Q_0+PC.RLOG                 "AMX/RAMX.0XT,DT/LONG,BMX/PC,ALU/A+B.RLOG,SHF/ALU,QK/SHF"
Q_0-D                       "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-K[]                     "AMX/RAMX.0XT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-LC                      "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-Q                       "AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_ACCEL&SYNC                "QK/ACCEL,ACF/SYNC"
Q_ALU                       "SHF/ALU,QK/SHF"
Q_ALU(FRAC)                 "SHF/ALU,QK/SHF.FL"
Q_ALU.LEFT                  "SHF/LEFT,QK/SHF"
Q_ALU.LEFT2                 "SHF/ALU.DT,DT/LONG,QK/SHF"
Q_ALU.LEFT3                 "QK/SHF,SHF/LEFT3"
Q_ALU.RIGHT                 "SHF/RIGHT,QK/SHF"
Q_ALU.RIGHT2                "SHF/RIGHT2,QK/SHF"
Q_D                         "QK/D"
Q_D(FRAC)(B)                "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,QK/SHF.FL"
Q_D+K[]                     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_D+K[]+1                   "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,QK/SHF"
Q_D+K[].LEFT                "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/LEFT,QK/SHF"
Q_D+LC                      "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF"
Q_D-K[]                     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_D-LC                      "RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_D-Q                       "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_D.0XT[]                   "RAMX/D,AMX/RAMX.0XT,DT/@1,ALU/A,SHF/ALU,QK/SHF"
Q_D.0XT[]+K[].LEFT          "RAMX/D,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B,SHF/LEFT,QK/SHF"
Q_D.0XT[].OR.PACK.FP        "RAMX/D,AMX/RAMX.0XT,DT/@1,BMX/PACKED.FL,ALU/OR,QK/SHF"
Q_D.AND.K[]                 "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_D.AND.K[].RIGHT           "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,QK/SHF"
Q_D.AND.K[].RIGHT2          "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT2,QK/SHF"
Q_D.AND.RC[]                "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,QK/SHF"
Q_D.ANDNOT.RC[]             "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_D.LEFT3                   "RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT3,QK/SHF"
Q_D.OR.K[]                  "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,QK/SHF"
```

```
Q_D.OR.RC[]              "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,QK/SHF"
Q_D.RIGHT                "RAMX/D,AMX/RAMX,ALU/A,SHF/RIGHT,QK/SHF"
Q_D.RIGHT2               "RAMX/D,AMX/RAMX,ALU/A,SHF/RIGHT2,QK/SHF"
Q_D.SXT[]                "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,QK/SHF"
Q_D.XOR.Q                "QK/SHF,ALU/XOR,AMX/RAMX,RAMX/D,BMX/RBMX,RBMX/Q,SHF/ALU"
Q_DEC.CON                "QK/DEC.CON"
Q_IB.BDEST               "IBC/BDEST,QK/ID,MCT/ALLOW.IB.READ"
Q_IB.DATA                "QK/ID,MCT/ALLOW.IB.READ"
Q_ID(SC)                 "CID/READ.SC,QK/ID"
Q_ID()                   "CID/READ.KMX,ID.ADDR/@1,QK/ID"
Q_K[]                    "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,QK/SHF"
Q_K[]+1                  "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,QK/SHF"
Q_K[].CTX                "KMX/@1,BMX/KMX,ALU/B,SHF/ALU.DT,DT/INST.DEP,QK/SHF"
Q_K[].RIGHT              "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT,QK/SHF"
Q_K[].RIGHT2             "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT2,QK/SHF"
Q_LA                     "AMX/LA,ALU/A,SHF/ALU,QK/SHF"
Q_LA+K[]                 "AMX/LA,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_LA+Q                   "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_LA-K[]                 "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_LA.AND.K[]             "AMX/LA,KMX/@1,BMX/KMX,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_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_RA[@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(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+D                    "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q+K[]                  "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q+K[]+1                "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,QK/SHF"
Q_Q+LC                   "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q+PC                   "RAMX/Q,AMX/RAMX,BMX/PC,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-K[]                  "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q-K[]-1                "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q-LC                   "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q-LC-1                 "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q-MASK-1               "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q.OXT[]-K[]            "RAMX/Q,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q.OXT[].LEFT           "RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/LEFT,QK/SHF"
Q_Q.OXT[].OR.D           "RAMX/Q,AMX/RAMX.OXT,DT/@1,RBMX/D,BMX/RBMX,ALU/OR,SHF/ALU,QK/SHF"
Q_Q.AND.K[]              "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_Q.AND.K[].RIGHT2       "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT2,QK/SHF"
Q_Q.AND.K[].RIGHT        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,QK/SHF"
Q_Q.AND.R[]              "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/LB,ALU/AND,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.ANDNOT.D             "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_Q.ANDNOT.K[]           "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
```

```
Q_Q.ANDNOT.RC[]          "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_Q.LEFT                 "QK/LEFT"
Q_Q.LEFT2                "QK/LEFT2"
Q_Q.OR.K[]               "RAMX/Q,AMX/RAMX,KMX/@1,RMX/KMX,ALU/OR,SHF/ALU,QK/SHF"
Q_Q.ORNOT.MASK           "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT,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_Q.XOR.K[]              "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR,SHF/ALU,QK/SHF"
Q_R(PRN).ANDNOT.Q        "SPO.AC/LOAD.LAB,SPO.ACN/PRN,AMX/LA,RBMX/Q,BMX/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(SC)                  "ALU/A,SHF/ALU,AMX/LA,SPO.AC/LOAD.LAB,SPO.ACN/SC,QK/SHF"
Q_R(SRC!1).AND.K[]       "SPO.AC/LOAD.LAB,SPO.ACN11/SRC.OR.1,AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_RC(SC)                 "ALU/B,SHF/ALU,BMX/LC,SPO/LOAD.LC.SC,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[]                    "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,QK/SHF"
Q_R[](FRAC)              "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,QK/SHF.FL"
Q_R[].AND.K[]            "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/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,BMX/KMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_R[].OR.K[]             "ALU/OR,AMX/LA,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/KMX,KMX/@2,QK/SHF"
Q_SC                     "ALU/B,BMX/KMX,KMX/SC,SHF/ALU,QK/SHF"
Q_SHF                    "QK/SHF"

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.SXT[].RIGHT     "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/RIGHT,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST"

R(PRN)_Q+D.RLOG          "ALU/A+B.RLOG,BMX/RBMX,RBMX/D,AMX/RAMX.OXT,DT/LONG,R(PRN)_ALU"
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+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)_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)_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+Q              "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+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[]MASK          "AMX/LA,BMX/MASK,ALU/@1,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R(PRN)_LC                "BMX/LC,ALU/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)_Q+K[].RLOG        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R(PRN)_ALU"
R(PRN)_Q-K[].RLOG        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B.RLOG,DT/LONG,R(PRN)_ALU"
R(PRN+1)_ALU             "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1"
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)_K[]            "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1"
R(PRN+1)_LA             "AMX/LA,ALU/A,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)_K[]               "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
R(SC)_LA                "AMX/LA,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_LC,R(SC)_ALU"
R(SC)_Q                 "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"

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(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(SRC!1)_ALU            "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.OR.1"
R(SRC!1)_D(B)           "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.OR.1"
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"
R(SRC)_LC               "BMX/LC,ALU/B,R(SRC)_ALU"
R(SRC)_Q                "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"

R6_D+K[].RLOG           "SPO.R/WRITE.RAB,SPO.RAB/R6,RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,SHF/ALU"
R6_LA+K[].RLOG          "AMX/LA,BMX/KMX,KMX/@1,ALU/A+B.RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/R6"
R6_LA-K[].RLOG          "AMX/LA,BMX/KMX,KMX/@1,ALU/A-B.RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/R6"

RC(SC)_0-LC             "ALU_0-LC,RC(SC)_ALU"
RC(SC)_ALU              "SHF/ALU,SPO/WRITE.RC.SC"
RC(SC)_ALU.RIGHT        "SPO/WRITE.RC.SC,SHF/RIGHT"
RC(SC)_D                "ALU_D,RC(SC)_ALU"
RC(SC)_Q                "ALU_Q,RC(SC)_ALU"

RC[]&VA_D+Q             "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0                  "AMX/RAMX.0XT,DT/LONG,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0+K[]+1            "AMX/RAMX.0XT,DT/LONG,KMX/@2,BMX/KMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0+LC+1             "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0+MASK+1           "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0+MASK+1.RIGHT2    "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0-D                "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,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.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[]_ALU.RIGHT2              "SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D                       "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D(B)                    "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D+K[]                   "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D-K[]                   "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.OXT[]                 "RAMX/D,AMX/RAMX.OXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.AND.K[]               "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.AND.MASK              "RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.ANDNOT.Q              "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.CTX                   "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.LEFT                  "RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.LEFT3                 "RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.OR.K[]                "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.OR.Q                  "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.ORNOT.K[]             "SPO.RC/@1,SPO.R/WRITE.RC,ALU/ORNOT,AMX/RAMX,RAMX/D,BMX/KMX,KMX/@2,SHF/ALU"
RC[]_D.SXT[]                 "RAMX/D,AMX/RAMX.SXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_K[]                     "KMX/@2,BMX/KMX,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_K[]+1                   "AMX/RAMX.OXT,DT/LONG,KMX/@2,BMX/KMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_K[].LEFT2               "KMX/@2,BMX/KMX,ALU/B,SHF/ALU.DT,DT/LONG,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_K[].LEFT3               "KMX/@2,BMX/KMX,ALU/B,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_K[].RIGHT2              "KMX/@2,BMX/KMX,ALU/B,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LA                      "AMX/LA,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LA+LB.CTX               "AMX/LA,BMX/LB,ALU/A+B,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LA-K[]                  "AMX/LA,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LA.AND.K[]              "ALU_LA.AND.K[@2],RC[@1]_ALU"
RC[]_LA.CTX                  "AMX/LA,ALU/A,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LB                      "BMX/LB,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LB.LEFT                 "BMX/LB,ALU/B,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LC                      "BMX/LC,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_NOT.Q                   "RAMX/Q,AMX/RAMX,ALU/NOTA,RC[@1]_ALU"
RC[]_PACK.FP                 "BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_PC                      "BMX/PC,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q                       "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q+1                     "ALU_Q+Q+1,RC[@1]_ALU"
RC[]_Q+K[]                   "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q+LC                    "ALU/A+B,RAMX/Q,AMX/RAMX,BMX/LC,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q+PC                    "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q+PC+1                  "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q-K[]                   "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q-LC                    "ALU/A-B,RAMX/Q,AMX/RAMX,BMX/LC,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q-MASK-1                "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.OXT[]                 "RAMX/Q,AMX/RAMX.OXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.AND.K[]               "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.ANDNOT.K[]            "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.LEFT                  "RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.LEFT3                 "RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/@1"
```

```
RC[]_Q.RIGHT                "RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.RIGHT2               "ALU_Q,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.SXT[]                "RAMX/Q,AMX/RAMX.SXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_RLOG.RIGHT             "BMX/0,MSC/READ.RLOG,ALU/B,SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/@1"

R[]&VA_LA+K[]               "AMX/LA,RMX/@2,BMX/RMX,ALU/A+B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]&VA_LA-K[]               "AMX/LA,RMX/@2,BMX/RMX,ALU/A-B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]&VA_LA-K[].RLOG          "AMX/LA,RMX/@2,BMX/RMX,ALU/A-B.RLOG,DT/LONG,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]&VA_Q-K[]                "RAMX/Q,AMX/RAMX,RMX/@2,BMX/RMX,ALU/A-B,VAK/LOAD,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0                       "SPO.R/WRITE.RAB,SPO.RAB/@1,AMX/RAMX.0XT,DT/LONG,ALU/A,SHF/ALU"
R[]_0+LB+1                  "AMX/RAMX.0XT,DT/LONG,BMX/LB,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-1                     "AMX/RAMX.0XT,DT/LONG,BMX/RMX,RMX/.1,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-D                     "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-K[]                   "AMX/RAMX.0XT,DT/LONG,RMX/@2,BMX/RMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-LB                    "AMX/RAMX.0XT,DT/LONG,BMX/LB,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-Q                     "AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,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.LEFT3               "SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/LEFT3"
R[]_ALU.RIGHT               "SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_ALU.RIGHT2              "SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/RIGHT2"
R[]_D                       "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,ALU/A,SHF/ALU"
R[]_D+K[]                   "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,RMX/@2,BMX/RMX,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[]_D-K[]                   "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,RMX/@2,BMX/RMX,ALU/A-B,SHF/ALU"
R[]_D-LC-1                  "ALU_D-LC-1,R[@1]_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.AND.K[]               "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/AND,AMX/RAMX,RAMX/D,BMX/RMX,RMX/@2,SHF/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[]_K[]                     "BMX/RMX,RMX/@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+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+K[]                  "AMX/LA,BMX/RMX,RMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA+K[]+1                "AMX/LA,BMX/RMX,RMX/@2,ALU/A+B+1,R[@1]_ALU"
R[]_LA+K[].RLOG             "AMX/LA,BMX/RMX,RMX/@2,ALU/A+B.RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
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+Q                    "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,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/RMX,RMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA-K[].RLOG             "AMX/LA,BMX/RMX,RMX/@2,ALU/A-B.RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA-MASK-1               "ALU/A-B-1,AMX/LA,BMX/MASK,SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/ALU"
R[]_LA-Q                    "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA.AND.K[]              "AMX/LA,BMX/RMX,RMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA.OR.D                 "AMX/LA,RBMX/D,BMX/RBMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_LA.ORNOT.MASK           "AMX/LA,BMX/MASK,ALU/ORNOT,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.Q                   "AMX/RAMX.OXT,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.OXT,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                       "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU"
R[]_Q+1                     "ALU_Q+Q+1,R[@1]_ALU"
R[]_Q+5                     "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/A+B+1,BMX/KMX,KMX/.4,AMX/RAMX,RAMX/Q,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+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/RAMX,BMX/LC,ALU/A+B,SHF/ALU"
R[]_Q-D                     "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU"
R[]_Q-D-1                   "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D,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,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A-B.RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_Q-LC                    "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU"
R[]_Q.AND.K[]               "ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/@1,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@2"
R[]_Q.ANDNOT.K[]            "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@2,SHF/ALU"
R[]_Q.OR.D                  "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/OR,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D,SHF/ALU"
R[]_Q.ORNOT.K[]             "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/KMX,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            "BMX/0,MSC/READ.RLOG,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1"

SC&STATE_STATE-R[](EXP)     "LAB_R[@1],AMX/LA,EBMX/AMX.EXP,MSC/LOAD.STATE,EALU/A-B,SMX/EALU,SCK/LOAD"
SC_0(A)                     "AMX/RAMX.OXT,DT/LONG,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
SC_0-K[]                    "BMX/KMX,KMX/@1,AMX/RAMX.OXT,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                        "RAMX/D,AMX/RAMX,ALU/A,SMX/ALU,SCK/LOAD"
SC_D(EXP)                   "RAMX/D,AMX/RAMX,ALU/A,SMX/ALU.EXP,SCK/LOAD"
SC_D(EXP)(A)                "RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
SC_D(EXP)(B)                "RBMX/D,BMX/RBMX,ALU/B,SMX/ALU.EXP,SCK/LOAD"
SC_D-K[]                    "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_D.OXT[]-K[]              "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_D.OXT[].XOR.K[]          "RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/KMX,KMX/@2,ALU/XOR,SC_ALU"
SC_D.AND.K[]                "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD"
SC_D.OR.K[]                 "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SMX/ALU,SCK/LOAD"
SC_D.SXT[]                  "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SMX/ALU,SCK/LOAD"
SC_EALU                     "SMX/EALU,SCK/LOAD"
SC_FE                       "SMX/FE,SCK/LOAD"
SC_K[]                      "KMX/@1,EBMX/KMX,EALU/B,SMX/EALU,SCK/LOAD"
SC_K[].ALU                  "KMX/@1,BMX/KMX,ALU/B,SMX/ALU,SCK/LOAD"
SC_LA                       "AMX/LA,ALU/A,SMX/ALU,SCK/LOAD"
SC_LA.AND.K[]               "AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD"
SC_LC(EXP)                  "BMX/LC,ALU/B,SMX/ALU.EXP,SCK/LOAD"
SC_NABS(SC-FE)              "EBMX/FE,EALU/NABS.A-B,SMX/EALU,SCK/LOAD"
SC_PSLADDR                  "SMX/EALU,EBMX/KMX,SCK/LOAD,KMX/.F,EALU/B"
```

```
SC_Q                    "RAMX/Q,AMX/RAMX,ALU/A,SMX/ALU,SCK/LOAD"
SC_Q(EXP)               "RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
SC_Q(EXP)(B)            "RBMX/Q,BMX/RBMX,ALU/B,SMX/ALU.EXP,SCK/LOAD"
SC_Q+K[]                "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/A+B,SMX/ALU,SCK/LOAD"
SC_Q-K[]                "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_Q.AND.K[]            "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/AND,SMX/ALU,SCK/LOAD"
SC_Q.OR.K[]             "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/OR,SMX/ALU,SCK/LOAD"
SC_Q.SXT[]              "RAMX/Q,AMX/RAMX.SXT,DT/@1,ALU/A,SMX/ALU,SCK/LOAD"
SC_RC[]                 "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SMX/ALU,SCK/LOAD"
SC_RC[](EXP)            "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SMX/ALU.EXP,SCK/LOAD"
SC_R[]                  "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SMX/ALU,SCK/LOAD"
SC_R[](EXP)             "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SMX/ALU.EXP,SCK/LOAD"
SC_R[].AND.K[]          "ALU/AND,AMX/LA,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/KMX,KMX/@2,SMX/ALU,SCK/LOAD"
SC_SC+1                 "EALU/A+1,SMX/EALU,SCK/LOAD"
SC_SC+EXP(Q)(A)         "EALU/A+B,EBMX/AMX.EXP,SMX/EALU,SCK/LOAD,AMX/RAMX,RAMX/Q"
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+SHF.VAL           "EALU/A+B,EBMX/SHF.VAL,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-SHF.VAL           "EBMX/SHF.VAL,EALU/A-B,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.OR.K[]            "KMX/@1,EBMX/KMX,EALU/OR,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.OR.K[]         "EALU/OR,EBMX/KMX,MSC/LOAD.STATE,SMX/EALU,SCK/LOAD,KMX/@1"
SD_NOT.SD               "SGN/NOT.SD"
SD_SS                   "SGN/SD.FROM.SS"
SS_0&SD_0               "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.OXT,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_FIRST             "STATE_K[ZERO]"                         ;EDITPC STATES
STATE_INNEROBJ          "STATE_K[.1]"                           ;MATCHC STATES
STATE_INNERSRC          "STATE_K[.3]"
STATE_K[]               "KMX/@1,EBMX/KMX,EALU/B,MSC/LOAD.STATE"
STATE_OUTER             "STATE_K[ZERO]"
STATE_PREDEC            "STATE_K[.80]"
STATE_Q(EXP)            "RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
STATE_SC.VIA.KMX        "MSC/LOAD.STATE,EALU/B,EBMX/KMX,KMX/SC"
STATE_SKPLONG           "STATE_K[.4]"                           ;SKPC STATES
STATE_STATE+1           "EALU/A+1,MSC/LOAD.STATE"
STATE_STATE+FE          "EBMX/FE,EALU/A+B,MSC/LOAD.STATE"
STATE_STATE+K[]         "KMX/@1,EBMX/KMX,EALU/A+B,MSC/LOAD.STATE"
```

```
STATE_STATE-FE                  "EBMX/FE,EALU/A-B,MSC/LOAD.STATE"
STATE_STATE-K[]                 "KMX/@1,EBMX/KMX,EALU/A-B,MSC/LOAD.STATE"
STATE_STATE.AN.SKPLONG          "STATE_STATE.ANDNOT.K[.4]"
STATE_STATE.AN.5TO0             "STATE_STATE.ANDNOT.K[.3F]"
STATE_STATE.AN.6TO4             "STATE_STATE.ANDNOT.K[.7F]"
STATE_STATE.AN.DESTDBL          "STATE_STATE.ANDNOT.K[.6]"
STATE_STATE.AN.NOTPREDEC        "STATE_STATE.ANDNOT.K[.7F]"
STATE_STATE.AN.PREDECZERO       "STATE_STATE.ANDNOT.K[.C0]"
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.ANDNOT.SHF.VAL      "MSC/LOAD.STATE,EBMX/SHF.VAL,EALU/ANDNOT"
STATE_STATE.OR.FE               "EALU/OR,EBMX/FE,MSC/LOAD.STATE"
STATE_STATE.OR.K[]              "KMX/@1,EBMX/KMX,EALU/OR,MSC/LOAD.STATE"
STATE_STATE.OR.ADJINP           "STATE_STATE.OR.K[.3]"
STATE_STATE.OR.DEST             "STATE_STATE.OR.K[.4]"
STATE_STATE.OR.DESTDBL          "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.MOVE             "STATE_STATE.OR.K[.50]"
STATE_STATE.OR.PATT1            "STATE_STATE.OR.K[.1]"
STATE_STATE.OR.PATT2            "STATE_STATE.OR.K[.2]"
SWAPD                           "DK/BYTE.SWAP"

VA_ALU                          "VAK/LOAD"
VA_D                            "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD"
VA_D+K[]                        "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD"
VA_D+LC                         "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD"
VA_D+Q                          "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD"
VA_D.0XT[]+Q                    "RAMX/D,AMX/RAMX.0XT,DT/@1,BMX/RBMX,ALU/A+B,VAK/LOAD"
VA_D.ANDNOT.K[]                 "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@1,ALU/ANDNOT,VAK/LOAD"
VA_K[]                          "KMX/@1,BMX/KMX,ALU/B,VAK/LOAD"
VA_LA                           "AMX/LA,ALU/A,VAK/LOAD"
VA_LA+D                         "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,VAK/LOAD"
VA_LA+K[]                       "AMX/LA,BMX/KMX,KMX/@1,ALU/A+B,VAK/LOAD"
VA_LA+K[]+1                     "AMX/LA,BMX/KMX,KMX/@1,ALU/A+B+1,VAK/LOAD"
VA_LA+PC                        "AMX/LA,BMX/PC,ALU/A+B,VAK/LOAD"
VA_LA+Q                         "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD"
VA_LA-D                         "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,VAK/LOAD"
VA_LA-K[]                       "AMX/LA,BMX/KMX,KMX/@1,ALU/A-B,VAK/LOAD"
VA_LA-K[]-1                     "AMX/LA,BMX/KMX,KMX/@1,ALU/A-B-1,VAK/LOAD"
VA_LA-Q                         "VAK/LOAD,ALU/A-B,AMX/LA,BMX/RBMX,RBMX/Q,SHF/ALU"
VA_LA.AND.LC                    "AMX/LA,BMX/LC,ALU/AND,VAK/LOAD"
VA_LA.ANDNOT.K[]                "AMX/LA,BMX/KMX,KMX/@1,ALU/ANDNOT,VAK/LOAD"
VA_LB+D.0XT                     "BMX/LB,ALU/A+B,AMX/RAMX.0XT,DT/BYTE,VAK/LOAD"
VA_PC                           "BMX/PC,ALU/B,VAK/LOAD"
VA_Q                            "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD"
VA_Q+D                          "VAK/LOAD,ALU/A+B,AMX/RAMX,BMX/RBMX,RAMX/Q,RBMX/D,SHF/ALU"
VA_Q+K[]                        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD"
VA_Q+LB                         "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B,VAK/LOAD"
VA_Q+LB.PC                      "RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD"
```

```
VA_Q+LC                         "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD"
VA_Q+PC                         "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD"
VA_Q-K[]                        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD"
VA_Q-LB                         "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A-B,VAK/LOAD"
VA_Q.ANDNOT.K[]                 "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,VAK/LOAD"
VA_RC[]                         "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,VAK/LOAD"
VA_R[]                          "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,VAK/LOAD"
VA_VA+4                         "PCK/VA+4"


.TOC    "       Macro definition      : Non-transfer macros"

B.FORK                          "LAB_R(SP1),QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/B.FORK"
BYTE                            "DT/BYTE"

C.FORK                          "SUB/SPEC,J/C.FORK"
CACHE.INVALIDATE                "MCT/INVALIDATE,VAK/NOP"
CALL                            "SUB/CALL"
CALL[]                          "CALL,J/@1"
CHK.FLT.OPR                     "MSC/CHK.FLT.OPR"
CHK.ODD.ADDR                    "MSC/CHK.ODD.ADDR"
CLK.UBCC                        "CCK/LOAD.UBCC"

CLR.FPD                         "MSC/CLR.FPD"
CLR.IB.COND                     "IBC/CLR.1-5.COND"
CLR.IB.OPC                      "IBC/CLR.0,IEK/ISTR"
CLR.IB.SPEC                     "IBC/CLR.1"
CLR.IB0-1                       "IBC/CLR.0.1,IEK/ISTR"
CLR.IB0-3                       "IBC/CLR.0-3"           ;DISCARD -11 INSTR & OPERAND
CLR.IB2-3                       "IBC/CLR.2.3"           ;11 MODE DISCARD ISTREAM OPERAND
CLR.IB2-5                       "IBC/CLR.1-5.COND"      ;2ND PART OF Q/D IMMEDIATE
CLR.NEST.ERR                    "MSC/CLR.NEST.ERR"
CLR.SD&SS                       "SGN/CLR.SD+SS"

E.FORK                          "SUB/SPEC,J/E.FORK"
EXCEPT.ACK                      "IEK/EACK"

FLUSH.IB                        "IBC/FLUSH,VAK/LOAD,IEK/ISTR"

G.FORK                          "SUB/SPEC,J/G.FORK"

INHIBIT.IB                      "MCT/MEM.NOP"
INTRPT.ACK                      "IEK/IACK"
INTRPT.STROBE                   "IEK/ISTR"
IRD                             "IRD0,CLK.UBCC,IRD1,SUB/SPEC,J/A.FORK"
IRD,11                          "LA_R(DST)&LB_R(SRC),D_LB.PC,VAK/LOAD,Q_IB.DATA,SC_K[.10],PCK/PC+N,MSC/IRD,SUB/SPEC,J/DP0"
IRD0                            "LA_R(SP2)&LB_R(SP1),D&VA_LB,SC_ALU(EXP),FE_LA(EXP),SS_ALU15"
IRD1                            "MSC/IRD,QK/ID,MCT/ALLOW.IB.READ,IBC/CLR.1-5.COND,PCK/PC+N"
```

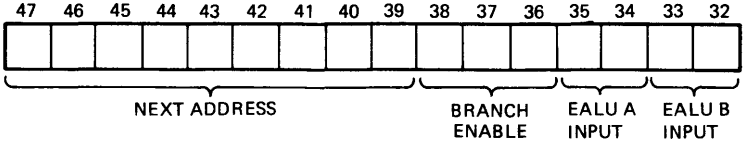
```
LOAD.ACC.CC             "MSC/LOAD.ACC.CC"
LOAD.IB                 "VAK/NOP,MCT/READ.V.NEWPC"
LOAD.IB.11              "VAK/NOP,MCT/READ.V.NEWPC"
LONG                    "DT/LONG"

MEMORY.NOP              "MCT/MEM.NOP"
MUL.0XT                 "SI/MUL+,SC_SC-K[.1],BEN/MUL"
MUL.1XT                 "SI/MUL-,SC_SC-K[.1],BEN/MUL"
MULM.DONE               "D_D.RIGHT2,SI/MUL-,INTRPT.STROBE"
MULP.DONE               "D_D.RIGHT2,SI/MUL+,INTRPT.STROBE"

POLY.DONE               "ACF/CONTROL,ACM/POLY.DONE"

RETURN0                 "SUB/RET,J/0"
RETURN1                 "SUB/RET,J/1"
RETURN10                "SUB/RET,J/10"
RETURN100               "SUB/RET,J/100"
RETURN10C               "SUB/RET,J/10C"
RETURN10E               "SUB/RET,J/10E"
RETURN12                "SUB/RET,J/12"
RETURN18                "SUB/RET,J/18"
RETURN1F                "SUB/RET,J/1F"
RETURN2                 "SUB/RET,J/2"
RETURN20                "SUB/RET,J/20"
RETURN24                "SUB/RET,J/24"
RETURN3                 "SUB/RET,J/3"
RETURN4                 "SUB/RET,J/4"
RETURN40                "SUB/RET,J/40"
RETURN60                "SUB/RET,J/60"
RETURN61                "SUB/RET,J/61"
RETURN8                 "SUB/RET,J/8"
RETURN9                 "SUB/RET,J/9"
RETURNF                 "SUB/RET,J/0F"
RETURN[]                "SUB/RET,J/@1"
```

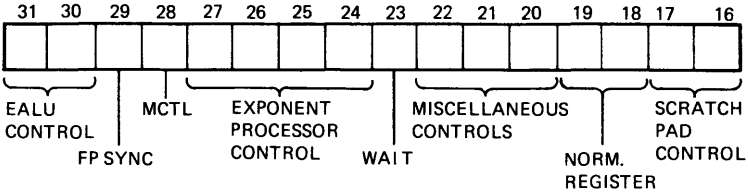
```
SET.CC(BYTE)                    "CCK/INST.DEP,DT/BYTE"
SET.CC(INST)                    "CCK/INST.DEP,DT/INST.DEP"
SET.CC(LONG)                    "CCK/INST.DEP,DT/LONG"
SET.CC(ROR)                     "CCK/ROR"
SET.CC(WORD)                    "CCK/INST.DEP,DT/WORD"
SET.FPD                         "MSC/SET.FPD"
SET.NEST.ERR                    "MSC/SET.NEST.ERR"
SET.PSL.C(AMX)                  "CCK/C_AMX0"
SET.V                           "CCK/SET.V"
SPEC                            "LAB_R(SP1),Q_IB.DATA,CLR.IB.COND,PC_PC+N,MCT/ALLOW.IB.READ,SUB/SPEC,J/C.FORK"
SPECG                           "LAB_R(SP1),Q_IB.DATA,CLR.IB.COND,PC_PC+N,MCT/ALLOW.IB.READ,SUB/SPEC,J/G.FORK"
START.IB                        "IBC/START"
STOP.IB                         "IBC/STOP"

TEST.TB.RCHK                    "MCT/TEST.RCHK,VAK/NOP"
TEST.TB.WCHK                    "MCT/TEST.WCHK,VAK/NOP"
TRAP.ACC()                      "ACF/TRAP,ACM/@1"

WORD                            "DT/WORD"
WRITE.DEST                      "LAB_R(SP1),QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/WRD"
WRITE.G.DEST                    "LAB_R(SP1), QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/WRG"
```

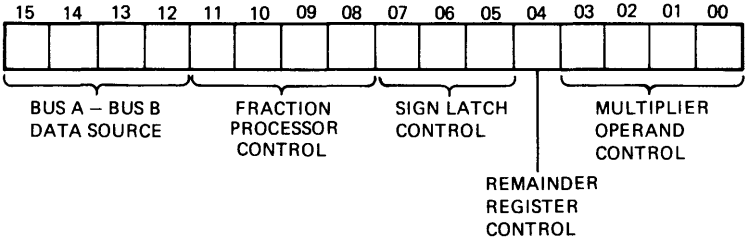
```
.TOC    "       Macro definition        : Branch enable macros"

AC.LOW?                         "BEN/INTERRUPT" ;,J3/3"
ACC.SYNC?                       "BEN/ACCEL"     ;,J3/3"
ACCEL?                          "BEN/ACCEL"
ALIGNED?                        "BEN/TB.TEST"   ;,J5/17"
ALU.N?                          "BEN/ALU"       ;,J4/07"
ALU1-0?                         "BEN/ALU1-0"
ALU?                            "BEN/ALU"

BCDSGN?                         "BEN/DECIMAL"   ;,J2/2"

C31?                            "BEN/C31"
CONSOLE.MODE?                   "BEN/PSL.MODE"  ;,J5/1B"

D(1)?                           "BEN/MUL"
D.B0?                           "BEN/D.BYTES"   ;,J4/0E"
D.B1?                           "BEN/D.BYTES"   ;,J4/0D"
D.B2?                           "BEN/D.BYTES"   ;,J4/0B"
D.BYTES?                        "BEN/D.BYTES"
D.NE.0?                         "BEN/SIGNS"     ;,J3/5" ;PREFERED FORM
D0?                             "BEN/D3-0"      ;,J4/0E"
D2-0?                           "BEN/D3-0"      ;,J4/08"
D2?                             "BEN/D3-0"      ;,J4/0B"
D3-0?                           "BEN/D3-0"
D31?                            "BEN/SIGNS"     ;,J3/6"
D3?                             "BEN/D3-0"      ;,J4/07"
DATA.TYPE?                      "BEN/DATA.TYPE"
DBL?                            "BEN/DATA.TYPE"

EALU.N?                         "BEN/EALU"      ;,J4/07"
EALU.Z?                         "BEN/EALU"      ;,J4/0B"
EALU?                           "BEN/EALU"
END.DP1?                        "BEN/END.DP1"

FPD?                            "BEN/LAST.REF"  ;,J4/07"

IB.TEST?                        "BEN/IB.TEST"
INT?                            "BEN/INTERRUPT"
INTERRUPT.REQ?                  "BEN/INTERRUPT" ;,J3/5"
IR0.C31?                        "BEN/ALU"
IR0?                            "BEN/ALU"       ;,J4/0D"
IR1?                            "BEN/IR2-1"     ;,J3/6"
IR2-1?                          "BEN/IR2-1"

LAST.REF?                       "BEN/LAST.REF"

MODE.LSS.ASTLVL?                "BEN/REI"       ;,J3/3"
MUL?                            "BEN/MUL"
```

```
NEST.ERR?                   "BEN/LAST.REF"  ;,J4/0B"

PC.MODES?                   "BEN/PC.MODES"
PSL.C?                      "BEN/PSL.CC"    ;,J4/0E"
PSL.CC?                     "BEN/PSL.CC"
PSL.MODE?                   "BEN/PSL.MODE"
PSL.N?                      "BEN/PSL.CC"    ;,J4/7"
PSL.V?                      "BEN/PSL.CC"    ;,J4/0D"
PSL.Z?                      "BEN/PSL.CC"    ;,J4/0B"
PTE.VALID?                  "BEN/TB.TEST"   ;,J5/0F"

Q31?                        "BEN/SIGNS"     ;,J3/3"
QUAD?                       "BEN/DATA.TYPE"

RLOG.EMPTY?                 "BEN/ALU1-0"    ;,J4/7"
ROR?                        "BEN/ROR"

SC.GT.0?                    "BEN/SC"
SC.NE.0?                    "BEN/MUL"       ;,J3/3"
SC?                         "BEN/SC"
SIGNS?                      "BEN/SIGNS"
SRC.PC?                     "BEN/SRC.PC"    ;COMP MODE, BEN ON SRC R = PC
SS?                         "BEN/EALU"      ;,J4/0E"
STATE(7)?                   "STATE7-4?"
STATE0?                     "BEN/STATE3-0"  ;,J4/0E"
STATE1-0?                   "BEN/STATE3-0"  ;,J4/0C"
STATE1?                     "BEN/STATE3-0"  ;,J4/0D"
STATE2?                     "BEN/STATE3-0"  ;,J4/0B"
STATE3-0?                   "BEN/STATE3-0"
STATE3?                     "BEN/STATE3-0"  ;,J4/07"
STATE4?                     "BEN/STATE7-4"
STATE5?                     "BEN/STATE7-4"
STATE6?                     "BEN/STATE7-4"
STATE7-4?                   "BEN/STATE7-4"

TB.TEST?                    "BEN/TB.TEST"

VA31-30?                    "BEN/PSL.MODE"  ;,J5/07"
VA31?                       "BEN/PSL.MODE"  ;,J5/0F"

Z?                          "BEN/Z"
ZONED?                      "BEN/DECIMAL"   ;,J2/1"
```

FPA CONTROL WORD FIELDS







TK-0513

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

.RTOL
.HEXADECIMAL

NAD/=0,9,39,+                   ;NEXT ADDRESS DEFAULT IS THE FOLLOWING
                                ;MICRO WORD
                                ;
BEN/=0,3,36,D                   ;BRANCH ENABLE

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

AMXC/=2,2,34,D                  ;EALU A INPUT


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

BMXC/=0,2,32,D                  ;EALU B INPUT


        NK=0                    ;ROM NORMALIZATION CONSTANT
        XR=1                    ;X REGISTER
        #80=2                   ;RESTORE EXCESS 80(16) NOTATION
        LB=3                    ;SEL LB TO EALU


EALUC/=0,2,30,D                 ;EALU CONTROLS
```

```
        A=0                     ;PASS AMUX
        A-B=1                   ;AMX MINUS BMX
        A+B=2                   ;AMX PLUS BMX
        #3FF=3                  ;EALU=3FF

FPSYNC/=0,1,29,D                ;SYNC TO CPU

        NOP=0
        FPS=1                   ;ASSERT FPSYNC


MCTL/=0,1,28,D


        NOP=0
        CNT=1                   ; START A MULTIPLY

EAC/=0,4,24,D                   ;EXPONENT PROCESSOR CONTROLS

        NOP=0
        LDXR=1                  ; POLY ARG EXP REG GETS EALU
        LDPR=2                  ; PRODUCT EXP REG GETS EALU
        PR.XR=3                 ; LOAD PR & XR
        LDB=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
        LDAB=0C                 ; LA_BUS A, LB_BUS B
        XR.AB=0D                ; LA_BUS A, LB_BUS B, XR_EALU
        PR.AB=0E                ; LA_BUS A, LB_BUS B, PR_EALU
        LD.ALL=0F               ;
```

```
WAIT/=0,1,23,D

        NOP=0                   ;
        WAIT=1                  ;ENABLE STALL

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

        NOP=0                   ;
        P.ADD=1                 ;POLY ADD FOR SIGN PROCESSOR
        MC=2                    ;SET ERROR BIT FOR RESRVD OPND
        RR.0=3                  ;CLEAR REMAINDER REGISTER
        CC=4                    ;ERROR CONDITIONS
        LDSX=5                  ;LOAD SIGN OF X FOR POLY
        IR0=6                   ;INSTRUCTION BRANCH 0
        IR1=7                   ;INSTRUCTION BRANCH 1

NRC/=3,2,18,D                   ;NORMALIZER REGESTER

        NOP=3                   ;
        SL=2                    ;SHIFT LEFT (NORMALIZE)
        LD=0                    ;LOAD BUS A, BUS B

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

        CPU=0                   ;REGISTER ADDRESS FROM CPU
        DPR.R=1                 ;SP2+1,PRN+1
        R.R=2                   ;SP1,SP2
        DP=3                    ;PRN+1, PRN
```

```
BSC/=8,4,12,D                   ; DATA SOURCE FOR BUS A AND BUS B

        INTH=3                  ;INTEGER PRODUCT HI TO BUS A
        NL=4                    ;NSHFL TO BUS A, BUS A TO BUS B
        NH=5                    ;NSHFH & EXP TO BUS A, A TO B
        PQ=6                    ;PROD/QUOTH TO BUS A, P/QL TO BUS B
        INTL=7                  ;INTEGER PRODL TO BUS A
        ID=8                    ;ID BUS TO BUS A, B
        LR=9                    ;IBUF DATA REGISTER TO BUS
        ID.RB=0A                ;ID TO BUS A, RB TO BUS B
        R=0B                    ;RA TO BUS A, RB TO BUS B
        FAL.X=0C                ;HARDWARE DETERMINED
        FAL.LH=0D               ;FALUL TO BUS A, FALUH TO BUS B
        FAL.HL=CE               ;FALUH TO BUS A, FALUL TO BUS B

FADC/=0C,4,8,D                  ;FRACTION PROCESSOR CONTROLS

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


SGNC/=0,3,5,D                   ;SIGN LATCH CONTROLS

        NOP=0                   ;SIGN LATCHES UNCHANGED
        LDSA=1                  ;BUS A(15) TO SA
        A.SNA=2                 ;IF SUB, SA<- NOT SA ; ELSE SA <- SA
        A.RES=3                 ;RESULT SIGN TO SA
```

```
        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/=6,4,0,D                   ;MULTIPLIER OPERAND CONTROLS

        NOP=6                   ;
        MC=1                    ;LOAD MULTIPLICAND
        MC1=2                   ;LOAD UPPER HALF OF MULTIPLICAND
        MC.P0=4                 ;LOAD DP LOW HALVES
        MC0=5                   ;LOAD LOWER HALF OF MULTIPLICAND
        LDR.R=8                 ;LOAD ALL FOR R-R
        MC1.MP1=0A              ;LOAD M'CAND INTEGER&MULTIPLIER HIGH FRACTION
        MC1=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      FLOAT H          IRBR1 L          IRBR0 L (OPCODE BRANCH)
;
;2      SWR H            SWR H            SWR H   (NORMALIZATION SHIFT WITHIN RANGE)
;
;3      RSV H            B=0 H            A=0 H   (ZEROES AND RESERVED OPERANDS)
;
;4      POLY DONE L      CPSYNC H         FLOAT H (SYNCHRONIZATION WITH CPU)
;
;5      (A OR B)=0 H     SUB*ED<2 H       SUB*DOUBLE*ED>8 H       (EXP. DIFF.)
;
;6      0                0                MUL/DIV DONE H
;
;7      0                [PR=0+PR<9>] L   PR<8> H (OVER/UNDERFLOW IN POLY)
;
       "TRANSFER MACRO DEFINITIONS"

AR_PROD                 "BSC/PQ,FADC/AR"
BFORK                   "MSC/IR1,NAD/100,WAIT/WAIT"
BUS_0                   "BSC/NL,EALUC/K3FF,AMXC/PR"
CPU_ANSH                "BMXC/NK,BSC/NH,AMXC/PR,EALUC/A"
CPU_ANSL                "BMXC/NK,BSC/NL,AMXC/PR,EALUC/A"
EALU_PR                 "AMXC/PR,EALUC/A"
EALU_PR-XR              "AMXC/PR,BMXC/XR,EALUC/A-B"
ERCH                    "MSC/CC"
FPA.IRD                 "SCR/R.R,FADC/R1,MSC/IR0,BSC/R,SGNC/LDS,OPLD/LDR.R.EAC/LDAB"
FPSYNC                  "FPSYNC/FPS"
IRD                     "NAD/0A3,EALUC/3,FADC/LD,SGNC/A.RES,BSC/NH,OPLD/INIT,MSC/RR.0"
LA&PR_LB                "BSC/NH,AMXC/LB,EALUC/A,SAC/PR.LA"
LA&PR_PR-K              "AMXC/PR,EALUC/A-B,BMXC/#80,BSC/NH,EAC/PR.LA"
LABSA_0                 "EALUC/K3FF,BMXC/NK,BSC/NH,EAC/LDA,SGNC/A.RES,AMXC/PR"
LA_0                    "EALUC/K3FF,BMXC/NK,BSC/NH,EAC/LDA,AMXC/PR"
LA_LB                   "AMXC/LB,BMXC/LB,EALUC/A,BSC/NH,EAC/LDA"
LA_PR                   "AMXC/PR,EALUC/A,BMXC/LB,BSC/NH,EAC/LDA"
LA_PR+K                 "AMXC/PR,EALUC/A+B,BMXC/#80,BSC/NH,EAC/LDA"
LA_PR-K                 "AMXC/PR,EALUC/A-B,BMXC/#80,BSC/NH,EAC/LDA"
LB_0                    "EALUC/K3FF,BMXC/NK,BSC/NL,EAC/LDB,SGNC/LDSB"
LB_PR                   "AMXC/PR,EALUC/A,BMXC/LB,BSC/NH,EAC/LDB"
```

```
LB_PR-K          "AMXC/PR,EALUC/A-B,BMXC/#80,BSC/NH,EAC/LDB"
LOAD.A0          "FADC/AR0,OPLD/MC0"
LOAD.A1          "FADC/AR1,EAC/LDA,SGNC/LDSA,OPLD/LDR.R"
LOAD.B           "FADC/BR,EAC/LDB,SGNC/LDSB,OPLD/MP"
LOAD.B0          "FADC/BR0,OPLD/MP0"
LOAD.B1          "FADC/BR1,EAC/LDB,SGNC/LDSB,OPLD/MCI.MP1"
LOAD.COEFH       "BSC/ID,FADC/BR1,EAC/LDB,SGNC/LDSB"
LOAD.COEFL       "BSC/ID,FADC/BR0"
LOAD.MR          "SGNC/LDS,FADC/R1,EAC/LDAB,OPLD/LDR.R"
LOAD.2DB         "BSC/R,SCR.DPR.R,FADC/R0"
MC0_NSHFL        "BSC/NL,OPLD/MC0"
MC1_NSHFH        "BSC/NH,OPLD/MC1"
MCCNT            "MCTL/CNT"
MC_BR            "BSC/FAL.HL,FADC/B,OPLD/MC"
MINIT            "OPLD/INIT"
NOP              "BEN/0"
NORM.PQ          "EAC/LDPR,SGNC/A.RES,BSC/NH,MSC/CC,EALUC/A+B,AMXC/PR,BMXC/NK"
NORM.SUM         "EAC/LDPR,SGNC/A.RES,BSC/NH,MSC/P.ADD,EALUC/A+B,AMXC/PR,BMXC/NK"
NR_AR            "FADC/A,BSC/FAL.HL,NRC/LD"
NR_BR            "FADC/B,BSC/FAL.HL,NRC/LD"
NR_FMD           "FADC/A.B,BSC/FAL.X,NRC/LD"
NR_PRCD          "BSC/PQ,NRC/LD"
NR_QUOT          "BSC/PQ,NRC/LD"
POLY.ADD         "MSC/P.ADD"
PR_0             "AMXC/LB,BMXC/LB,EALUC/A-B,EAC/LDPR"
PR_COND          "AMXC/COND,EALUC/A,EAC/LDPR"
PR_K             "BMXC/#80,EALUC/K3FF,EAC/LDPR"
PR_LA            "AMXC/LA,EALUC/A,EAC/LDPR"
PR_LA+K          "AMXC/LA,EALUC/A+B,BMXC/#80,EAC/LDPR"
PR_LA+LB         "AMXC/LA,BMXC/LB,EALUC/A+B,EAC/LDPR"
PR_LA+NROM       "AMXC/LA,BMXC/NA,EALUC/A+B,EAC/LDPR"
PR_LB            "AMXC/LB,EALUC/A,EAC/LDPR"
```

TRANSFER MACRO DEFINITIONS

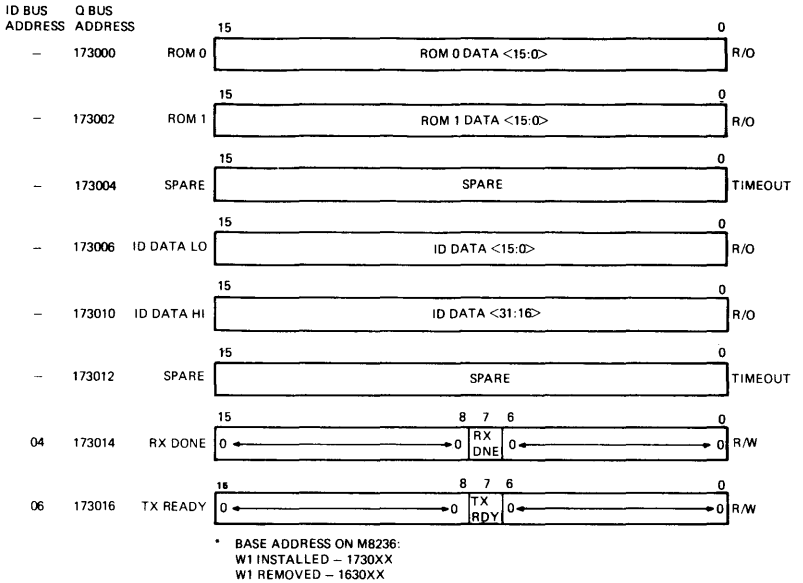
```
PR_LB+K                 "AMXC/LB,EALUC/A+B,BMXC/#80,EAC/LDPR"
PR_LB-K                 "AMXC/LB,EALUC/A-B,BMXC/#80,EAC/LDPR"
PR_LB-XR                "AMXC/LB,BMXC/XR,EALUC/A-B,EAC/LDPR"
PR_PR+K                 "AMXC/PR,BMXC/#80,EALUC/A+B,EAC/LDPR"
PR_PR+NROM              "AMXC/PR,BMXC/NR,EALUC/A+B,EAC/LDPR"
PR_PR+XR                "AMXC/PR,BMXC/XR,EALUC/A+B,EAC/LDPR"
PR_PR-K                 "AMXC/PR,BMXC/#80,EALUC/A-B,EAC/LDPR"
PR_UNDR                 "EAC/LDPR,EALUC/#3FF"
PR_XR+LB                "EAC/LDPR,AMXC/LB,BMXC/XR,EALUC/A+B"
SR_NR                   "LRR/LD.RS"
RSV                     "MSC/M0"
SA_SA.XOR.SUB           "SGNC/A.SNA"
SA_SA.XOR.SX            "SGNC/A.XOR.X"
SA_SB                   "SGNC/A.B"
SA_SR                   "SGNC/A.RES"
SX_SA                   "MSC/LDSX"
WAIT                    "WAIT/WAIT"
XR_LA                   "AMXC/LA,EALUC/A,EAC/LDXR"
```

"BRANCH MACRO DEFINITIONS"

```
(A.OR.B).0?             "BEN/5"
CPSYNC?                 "BEN/4"
DIV.DONE?               "BEN/6"
ERROR?                  "BEN/7"
EXP.DIFF?               "BEN/5"
FLOAT?                  "BEN/4"
MUL.DONE?               "BEN/6"
OPCODE?                 "BEN/1"
SWR?                    "BEN/2"
ZEROES?                 "BEN/3"
POLY.DONE?              "BEN/4"
```

CHAPTER 4
CONSOLE

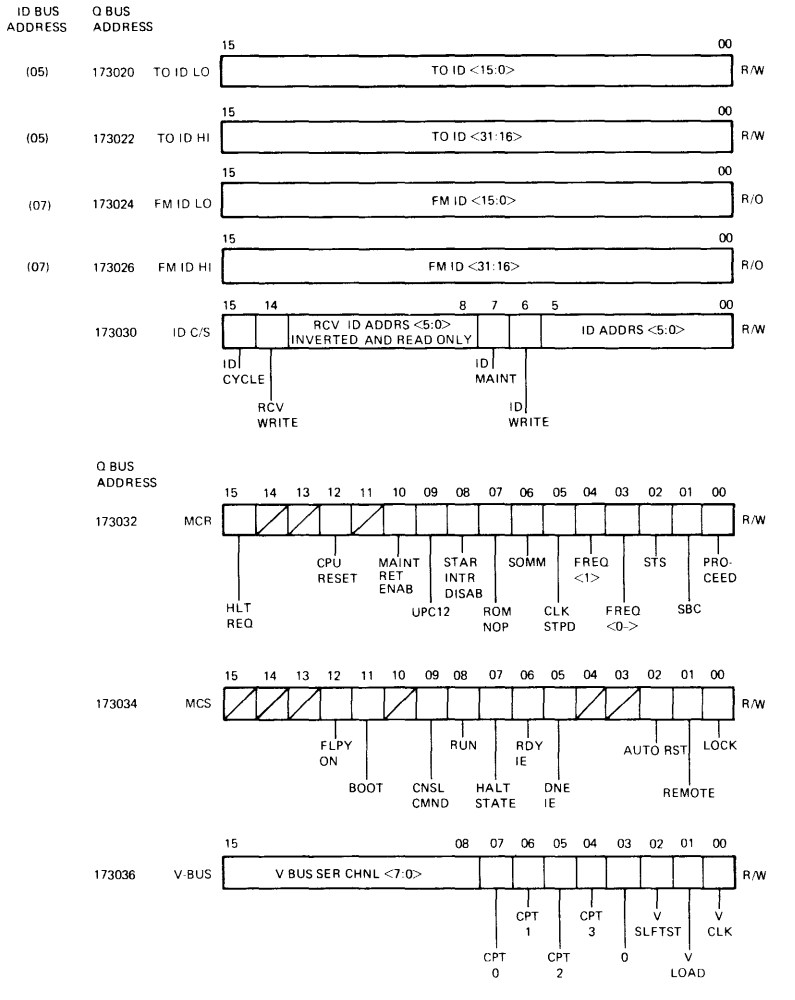
CIB Q-BUS REGISTERS (LOWER)



* BASE ADDRESS ON M8236:
W1 INSTALLED – 1730XX
W1 REMOVED – 1630XX

TK-0204

CIB Q-BUS REGISTERS (UPPER)



EXPLANATION OF VERSION NUMBERS FOR CONSOLE BOOTING

When WCS has been reloaded, the console terminal prints out revision status, for example:

```
VER:  PCS=Ø1 WCS=ØE-1Ø FPLA=ØE CON=VØ7-ØØ-L  KE78Ø PRESENT
```

The PCS code refers to the revision number of the programmed control store (ROM). The WCS (writable control store) code contains two numbers. The primary version number (in this case, ØE) refers to the FPLA number that is required for this WCS version. The secondary version number (in this case, 1Ø) refers to the version of WCS that has been loaded.

The FPLA (field programmable logic array) code refers to the FPLA chip revision that is currently installed in the VAX-11/78Ø CPU. This chip causes the microprocessor to retrieve microwords from WCS instead of from PCS when specific locations are addressed.

The CON (console) code refers to the revision number of the console software that has been loaded into the LSI-11 memory.

Two types of mismatch may occur. If the WCS revision does not match the FPLA revision, the console program issues a warning. If the WCS revision does not match the PCS revision, however, the mismatch is fatal.

CONSOLE HELP FILE

```
VAX-11/780 CONSOLE HELP FILE    REV. 8  March 29, 1982

SYNTAX:   ALL COMMANDS ARE TERMINATED BY CARRIAGE RETURN.

        'EXAMINE' AND 'DEPOSIT' <QUAL> SWITCHES FOR ADDRESS SPACE :
                '/P'  = PHYSICAL MEMORY(THE DEFAULT)
                '/V'  = VIRTUAL MEMORY
                '/I'  = INTERNAL(PROCESSOR) REGISTERS
                '/G'  = GENERAL REGISTERS 0 THRU F(R0 THRU PC)
                '/VB' = VBUS REGISTERS
                '/ID' = IDBUS REGISTERS
        'EXAMINE' AND 'DEPOSIT' <QUAL> SWITCHES FOR DATA-LENGTH :
                '/B'  = BYTE (8 BITS)
                '/W'  = WORD (2 BYTES)
                '/L'  = LONGWORD (2 WORDS)
                '/Q'  = QUADWORD (4 WORDS)
        <ADDR> IS A <NUMBER>, OR ONE OF THE FOLLOWING SYMBOLIC ADDRESS
                'R0,R1,R2,.....,R11,AP,FP,SP,PC'  (GENERAL REGISTERS)
                'PSL' = PROCESSOR STATUS WORD
                '*'   = LAST ADDRESS
                '+'   = ADDRESS FOLLOWING 'LAST'(*) ADDRESS
                '-'   = ADDRESS PRECEEDING 'LAST'(*) ADDRESS
                '@'   = USES LAST EXAMINE/DEPOSIT DATA FOR ADDRESS
        <NUMBER> = STRING OF DIGITS IN CURRENT DEFAULT RADIX,
        OR STRING OF DIGITS PREFIXED WITH A DEFAULT RADIX
        OVERRIDE (%O FOR OCTAL, %X FOR HEX).

'BOOT'                                  -BOOTS THE CPU FROM DEFAULT DEVICE
'BOOT <DEVNAM>'                         -TAKES THE FIRST THREE ALPHANUMERIC
                                         CHARS OF <DEVNAM>, AND EXECUTES THE
                                         INDIRECT FILE '<DEVNAM>BOO.CMD'
'CLEAR STEP'                            -ENABLE NORMAL (NO STEP) MODE
'CLEAR SOMM'                            -CLEAR 'STOP ON MICRO-MATCH' ENABLE.
                                         NOTE: ID REGISTER 21 IS THE
                                         MICRO-MATCH REGISTER.
'CONTINUE'                              -ISSUES A CONTINUE TO THE ISP
'DEPOSIT[/<SWITCH(ES)>] <ADDR> <DATA>'  -DEPOSIT <DATA> TO <ADDRESS>
'ENABLE DX1:'                           -ENABLES CONSOLE SOFTWARE TO ACCESS
                                         FLOPPY DRIVE 1 ON THOSE SYSTEMS WITH
                                         DUAL FLOPPIES.
'EXAMINE[/<SWITCH(ES)>] <ADDR>  '       -DISPLAY CONTENTS OF <ADDRESS>
'EXAMINE IR'                            -EXAMINE INSTRUCTION REG(IR). DISPLAYS
                                         OP-CODE, SPECIFIER, EXECUTION POINT
                                         COUNTER
'HALT'                                  -HALTS THE ISP
'HELP'                                  -PRINTS THIS FILE
'INITIALIZE'                            -INITIALIZES THE CPU
'LINK'                                  -CAUSES CONSOLE TO BEGIN COMMAND
                                         LINKING. CONSOLE PRINTS REVERSED
                                         PROMPT TO INDICATE LINKING. ALL
                                         COMMANDS TYPED BY USER WHILE LINKING
                                         ARE STORED IN AN INDIRECT COMMAND
                                         FILE FOR LATER EXECUTION. CONTROL-C
                                         TERMINATES LINKING.(SEE PERFORM)


'LOAD[/START:<ADDR>] <FILENAME>'        -LOAD FILE TO MAIN MEMORY, STARTING AT
                                         ADDRESS 0, OR <ADDR> IF SPECIFIED
'LOAD/WCS <FILENAME>'                   -LOAD FILE SPECIFIED TO WCS
'NEXT <NUMBER>'                         -<NUMBER> STEP CYCLES ARE DONE, TYPE
```

CONSOLE HELP FILE (CONT)

```
                                        OF STEP DEPENDS ON LAST 'SET STEP'
                                        COMMAND
'PERFORM'                               -EXECUTE A FILE OF LINKED COMMANDS
                                        PREVIOUSLY GENERATED VIA A 'LINK'
                                        COMMAND.
'QCLEAR <ADDRESS>'                      -DOES A QUAD CLEAR TO <ADDRESS>,WHICH
                                        IS FORCED TO A QUAD WORD BOUNDARY.
                                        (CLEARS ECC ERRORS)
'REBOOT'                                -CAUSES A CONSOLE SOFTWARE RELOAD
'REPEAT <ANY-CONSOLE-COMMAND>'          -CAUSES THE CONSOLE TO REPEATEDLY
                                        EXECUTE THE <CONSOLE-COMMAND>, UNTIL
                                        STOPPED BY A CONTROL-C (^C).
'SET CLOCK SLOW'                        -SET CPU CLOCK FREQ TO SLOW.
'SET CLOCK FAST'                        -SET CPU CLOCK FREQ TO FAST
'SET CLOCK NORMAL'                      -SET CPU CLOCK FREQ TO NORMAL
'SET DEFAULT <OPTION>[,...,<OPTION>]'   -SET CONSOLE DEFAULTS
                                        NOTE: <OPTIONS> ARE:
                                        OCTAL,HEX,PHYSICAL,VIRTUAL,INTERNAL
                                        GENERAL,VBUS,IDBUS,BYTE,WORD,LONG,QUAD
'SET RELOCATION:<NUMBER>'               -PUT <NUMBER> INTO CONSOLE RELOCATION
                                        REGISTER. RELOCATION REGISTER IS
                                        ADDED TO EFFECTIVE ADDRESS OF
                                        PHYSICAL AND VIRTUAL EXAMINES AND
                                        DEPOSITS.
'SET SOMM'                              -SET 'STOP ON MICRO-MATCH' ENABLE
'SET STEP BUS'                          -ENABLE SINGLE BUS CYCLE CLOCK MODE
'SET STEP INSTRUCTION'                  -ENABLES SINGLE INSTRUCTION MODE
'SET STEP STATE'                        -ENABLE SINGLE TIME STATE CLOCK MODE
'SET TERMINAL FILL:<NUMBER>'            -SET FILL COUNT FOR # OF BLANKS
                                        WRITTEN TO THE TERMINAL AFTER <CRLF>
                                        -PUT CONSOLE TERMINAL INTO 'PROGRAM
                                        I/O' MODE
'SHOW'                                  -SHOWS CONSOLE AND CPU STATE
'SHOW VERSION'                          -SHOWS VERSIONS OF MICROCODE AND
                                        CONSOLE
'START <ADDRESS>'                       -INITIALIZES THE CPU,DEPOSITS<ADDRESS>
                                        TO PC, ISSUES A CONTINUE TO THE ISP.
'TEST'                                  -RUNS MICRO-DIAGNOSTICS
'TEST/COM'                              -LOADS MICRO-DIAGNOSTICS, AWAITS
                                        COMMANDS
'UNJAM'                                 -UNJAMS THE SBI
'WCS'                                   -CALLS MICRO-DEBUGGER. WCS MICRO-
                                        DEBUGGER FLOPPY MUST BE INSERTED
                                        IN CS1. ELSE, "FILE NOT FOUND" ERROR.
                                        (FOR DEBUGGER HELP, INSERT WCS DEBUG
                                        FLOPPY, THEN TYPE '@WCSMON.HLP')
'WAIT DONE'                             -WHEN EXECUTED FROM AN INDIRECT
                                        COMMAND FILE, THIS COMMAND WILL CAUSE
                                        COMMAND FILE EXECUTION TO STOP UNTIL:
                                          A) A 'DONE' SIGNAL IS RECEIVED FROM
                                             THE PROGRAM RUNNING IN THE VAX
                                             (COMMAND FILE EXECUTION WILL
                                             CONTINUE), OR
                                          B) THE VAX-11/780 HALTS, OR OPER-
                                             ATOR TYPES A CONTROL-C (^C :
                                             COMMAND FILE EXECUTION WILL
                                             TERMINATE).
'^P'(CONTROL-P)                         -PUT CONSOLE TERMINAL INTO 'CONSOLE
                                        I/O' MODE
                                        (UNLESS MODE SWITCH IN 'DISABLE')
'@<FILENAME>'                           -PROCESS AN INDIRECT COMMAND FILE
```

CONSOLE ABBREVIATION RULES

```
VAX-11/780 CONSOLE ABBREVIATION RULES   REV-6   12-APR-79
 THIS FILE SHOWS THE SHORTEST UNIQUE COMMAND AND QUALIFIER STRINGS.

        COMMAND                     SHORTEST ABBREVIATION RECOGNIZED
 'BOOT'                                     'B'
 'CLEAR SOMM'                               'CL SO'
 'CLEAR STEP'                               'CL S'
 'CONTINUE'                                 'C'
 'DEPOSIT <ADDRESS> <DATA>'                 'D <ADDRESS> <DATA>'
 'ENABLE DX1:'                              'EN DX1:'
 'EXAMINE <ADDRESS>'                        'E <ADDRESS>'
 'HALT'                                     'H'
 'HELP'                                     'HE'
 'INITIALIZE'                               'I'
 'LINK'                                     'LI'
 'LOAD <FILENAME>'                          'L <FILENAME>'
 'NEXT <NUMBER>'                            'N <NUMBER>'
 'PERFORM'                                  'P'
 'QCLEAR <ADDRESS>'                         'Q <ADDRESS>'
 'REBOOT'                                   'REB'
 'REPEAT <CONSOLE-COMMAND>'                 'R <CONSOLE-COMMAND>'
 'SET CLOCK FAST'                           'SE C F'
 'SET CLOCK SLOW'                           'SE C S'
 'SET CLOCK NORMAL'                         'SE C N'
 'SET RELOCATION:<NUMBER>'                  'SE R:<NUMBER>'
 'SET SOMM'                                 'SE SO'
 'SET STEP BUS'                             'SE S B'
 'SET STEP STATE'                           'SE S S'
 'SET STEP INSTRUCTION'                     'SE S I'
 'SET TERMINAL FILL:<NUMBER>'               'SE T F:<NUMBER>'
 'SET TERMINAL PROGRAM'                     'SE T PR'
 'SET DEFAULT <OPTION-LIST>'                'SE D <OPTION-LIST>'
 'SHOW'                                     'SH'
 'SHOW VERSION'                             'SH V'
 'START <ADDRESS>'                          'S <ADDRESS>'
 'TEST'                                     'T'
 'UNJAM'                                    'U'
 'WAIT DONE'                                'WA D'
 'WCS'                                      'W'
 '@<FILENAME>'                              '@<FILENAME>'

        QUALIFIERS                  SHORTEST ABBREVIATION RECOGNIZED
 /BYTE                                      /B
 /WORD                                      /W
 /LONG                                      /L
 /QUAD                                      /Q
 /OCTAL                                     /O
 /HEX                                       /H
 /PHYSICAL                                  /P
 /VIRTUAL                                   /V
 /INTERNAL                                  /I
 /GENERAL                                   /G
 /VBUS                                      /VB
 /IDBUS                                     /ID
 /WCS                                       /WC
 /NEXT:<NUMBER>                             /N:<NUMBER>
 /COMMAND                                   /C
 /START:<ADDRESS>                           /S:<ADDRESS>
```

ERROR MESSAGE HELP FILE

```
!VAX-11/780 ERROR MESSAGE HELP FILE     REV-4   12-AR-79
! THIS FILE LISTS ALL THE POSSIBLE CONSOLE ERROR MESSAGES, INDICATING
! CAUSE, AND POSSIBLE CORRECTIVE PROCEDURES (IF NOT SELF-EXPLANATORY)
!====================================
! SYNTACTIC ERRORS :                !
!====================================================================
! ?'<TEXT-STRING>' IS INCOMPLETE    !   THE <TEXT-STRING> IS NOT A COMPLETE
!                                   !   CONSOLE COMMAND
!--------------------------------------------------------------------
! ?'<TEXT-STRING>' IS INCORRECT     !   THE <TEXT-STRING> IS NOT RECOGNIZED
!                                   !   AS A VALID COMMAND
!--------------------------------------------------------------------
! ?FILE NAME ERR                    !   A <FILE-NAME> GIVEN WITH A COMMAND
!                                   !   CAN NOT BE TRANSLATED TO RAD50.
!                                   !   (<FILE-NAME> IS INVALID.)
!--------------------------------------------------------------------
! ?IND-COM ERR                      !   THE CONSOLE DETECTED AN ERROR IN THE
!                                   !   FORMAT OF AN INDIRECT COMMAND FILE.
!                                   !   POSSIBLE ERRORS ARE: 1) MORE THAN 80
!                                   !   CHARACTERS IN AN INDIRECT COMMAND LINE
!                                   !   OR  2) A COMMAND LINE DID NOT END
!                                   !   WITH A CARRIGE RETURN AND LINE FEED.
!====================================--------------------------------------
! COMMAND-GENERATED ERRORS :        !
!====================================--------------------------------------
! ?FILE NOT FOUND                   !   A <FILE-NAME> GIVEN WITH A 'LOAD'
!                                   !   OR '@' COMMAND DOES NOT MATCH ANY
!                                   !   FILE ON THE CURRENTLY LOADED FLOPPY
!                                   !   DISC. CAN ALSO BE GENERATED BY
!                                   !   'HELP','BOOT', OR AN ATTEMPTED WCS
!                                   !   LOAD IF HELP FILE, BOOT FILE, OR WCS
!                                   !   FILE IS MISSING FROM FLOPPY.
!--------------------------------------------------------------------
! ?NO CPU RESPONSE                  !   CONSOLE TIMED-OUT WAITING FOR A
!                                   !   RESPONSE FROM CPU. (RETRY. INDICATES
!                                   !   POSSIBLE CPU-RELATED HARDWARE FAULT.)
!--------------------------------------------------------------------
! ?CPU NOT IN CONSOLE WAIT LOOP,    !   A CONSOLE COMMAND REQUIRING ASSISTANCE
!                 COMMAND ABORTED   !   FROM THE CPU WAS ISSUED WHILE THE CPU
!                                   !   WAS NOT IN THE CONSOLE SERVICE LOOP.
!                                   !   (HALT CPU; RE-ISSUE COMMAND.)
!--------------------------------------------------------------------
! ?CPU CLK STOP, COMMAND ABORTED    !   A CONSOLE COMMAND THAT REQUIRES THE
!                                   !   CPU CLOCK TO BE RUNNING WAS ISSUED
!                                   !   WHILE THE CLOCK WAS STOPPED.
!                                   !   (CLEAR STEP MODE; RE-ISSUE COMMAND.)
!--------------------------------------------------------------------
! ?CANT DISABLE BOTH FLOPPIES,      !   AN ATTEMPT WAS MADE TO DISABLE BOTH
!                 FUNCTION ABORTED  !   THE REMOTE AND LOCAL FLOPPY.
!--------------------------------------------------------------------
!====================================--------------------------------------
! FLOPPY-GENERATED ERRORS :         !
!====================================--------------------------------------
! ?FLOPPY ERR, CODE=X               !   THE CONSOLE FLOPPY DRIVER DETECTED
!                                   !   AN ERROR. CODES ARE AS FOLLOWS:
!                                   !   (CODES ALWAYS PRINTED IN HEX RADIX)
!                                   !   CODE=1  FLOPPY HARDWARE ERROR
!                                   !           (CRC,PARITY,ETC)
!                                   !   CODE=2  FILE NOT FOUND
!                                   !   CODE=3  FLOPPY DRIVER QUEUE OVERFLOW
!                                   !   CODE=4  CONSOLE SOFTWARE REQUESTED
!                                   !           AN ILLEGAL SECTOR NUMBER
!--------------------------------------------------------------------
```

ERROR MESSAGE HELP FILE (CONT)

?FLOPPY NOT READY	THE CONSOLE FLOPPY DRIVE FAILED TO BECOME READY WHEN BOOTING. (RETRY.)
?NO BOOT ON FLOPPY	CONSOLE ATTEMPTED TO BOOT FROM A FLOPPY THAT DOES NOT CONTAIN A VALID BOOT BLOCK. (CHANGE FLOPPY DISK.)
?FLOPPY ERROR ON BOOT	A FLOPPY ERROR WAS DETECTED WHILE ATTEMPTING A CONSOLE BOOT. (RETRY)
MICRO-ROUTINE ERRORS :	
?MIC-ERR ON FUNCTION	A MICRO-ERROR OCCURRED IN THE CPU WHILE SERVICING A CONSOLE REQUEST. SBI ERROR REGISTERS ARE DUMPED AFTER THIS MESSAGE IS PRINTED. (ACTION DEPENDANT ON ERROR.)
?INT-REG ERR	A MICRO-ERROR OCCURRED WHILE ATTEMPTING TO REFERENCE A CPU INTERNAL (PROCESSOR) REGISTER. AN ILLEGAL ADDRESS 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.
?MEM-MAN FAULT,CODE=XX	A VIRTUAL EXAMINE OR DEPOSIT CAUSED AN ERROR IN THE MEMORY MANAGEMENT MICRO-ROUTINE. 'XX' IS A ONE BYTE ERROR CODE RETURNED BY THE ROUTINE, WITH THE FOLLOWING BIT ASSIGNMENTS: BIT 0 = LENGTH VIOLATION(BITS NUMBERED FROM RIGHT) BIT 1 = FAULT WAS ON A PTE REFERENCE BIT 2 = WRITE OR MODIFY INTENT BIT 3 = ACCESS VIOLATION BITS 4 THRU 7 SHOULD BE IGNORED
CPU FAULT-GENERATED ERRORS :	
?INT-STK INVLD	THE CPU HALTED BECAUSE THE INTERRUPT STACK WAS MARKED INVALID.
?CPU DBLE-ERR HLT	A MACHINE CHECK OCCURRED BEFORE A PREVIOUS MACHINE CHECK HAD BEEN HANDLED, CAUSING THE CPU TO EXECUTE A 'DOUBLE ERROR' HALT. (EXAMINE IPR REG 30-3F (HEX); DATA INDICATES CAUSE OF MACHINE CHECK .)
?ILL I/E VEC	THE CPU DETECTED AN ILLEGAL INTERRUPT/EXCEPTION VECTOR.
?NO USR WCS	CPU DETECTED AN INTERRUPT/EXCEPTION VECTOR TO USER WCS AND NO USER WCS EXISTS.
?CHM ERR	A CHANGE MODE INSTRUCTION WAS ATTEMPTED FROM THE INTERRUPT STACK.

ERROR MESSAGE HELP FILE (CONT)

INT PENDING	THIS IS NOT ACTUALLY AN ERROR, BUT INDICATES THAT AN ERROR WAS PENDING AT THE TIME THAT A CONSOLE-REQUESTED HALT WAS PERFORMED. CONTINUE CPU TO CLEAR INTERRUPT.
?MICRO-MACHINE TIME OUT	INDICATES THAT THE VAX-11/780 MICRO-MACHINE HAS FAILED TO STROBE INTERRUPTS WITHIN THE MAXIMUM TIME PERIOD ALLOWED.
VERSION MISMATCH ERRORS :	
?WARNING-WCS & FPLA VER MISMATCH	THE MICROCODE IN WCS IS NOT COMPATIBLE WITH THE FPLA. THIS MESSAGE IS PRINTED ON EACH ISP START OR CONTINUE,BUT NO OTHER ACTION TAKEN BY CONSOLE.
?FATAL-WCS & PCS VER MISMATCH	THE MICROCODE IN PCS IS NOT COMPATIBLE WITH THAT IN WCS. ISP START AND CONTINUE ARE DISABLED BY CONSOLE.
?REMOTE ACCESS NOT SUPPORTED	PRINTED WHEN CONSOLE MODE SWITCH ENTERS A 'REMOTE' POSITION, AND THE REMOTE SUPPORT SOFTWARE ROUTINES ARE NOT INCLUDED IN THE CONSOLE.
CONSOLE-GENERATED ERRORS :	
?TRAP-4, RESTARTING CONSOLE	THE CONSOLE TOOK A TIME-OUT TRAP. CONSOLE WILL RESTART.
?UNEXPECTED TRAP MOUNT CONSOLE FLOPPY, THEN TYPE ^C	CONSOLE TRAPPED TO AN UNUSED VECTOR. CONSOLE REBOOTS WHEN ^C TYPED.
?Q-BLKD'	CONSOLE'S TERMINAL OUTPUT QUEUE IS BLOCKED. CONSOLE WILL REBOOT.

CONSOLE REMOTE ACCESS HELP FILE

```
VAX-11/780 CONSOLE - REMOTE ACCESS HELP FILE REV-02 26-JUL-78

 'ENABLE TALK'           -ESTABLISH TERMINAL TO TERMINAL COMMUNICATION
                          BETWEEN LOCAL AND REMOTE TERMINAL. KEYS
                          STRUCK ON ONE TERMINAL ARE PRINTED ON THE
                          OTHER. CONTROL-P TERMINATES TALK.
 'ENABLE ECHO'           -CAUSES CHARACTERS TYPED IN TALK MODE TO BE
                          ECHOED BACK TO THE ORIGINATING TERMINAL.
 'ENABLE LOCAL COPY'     -CAUSES THE LOCAL TERMINAL TO GET A COPY OF
                          OF OUTPUT BEING SENT TO REMOTE TERMINAL.
 'ENABLE LOCAL CONTROL '-ALLOWS LOCAL TERMINAL TO CONTROL SYSTEM WHEN
                          CONSOLE SWITCH IS IN REMOTE POSITION(S). DIS-
                          ABLED BY A CONTROL-P FROM THE REMOTE TERMINAL.
 'ENABLE CARRIER ERROR '-CAUSE CONSOLE TO PRINT '?CARRIER LOST' WHEN A
                          LOSS OF CARRIER IS DETECTED AT REMOTE INTERFACE.
 'DISABLE ECHO'          -INHIBITS ECHO OF CHARACTERS TYPED IN TALK MODE.
 'DISABLE LOCAL COPY'    -DISABLE LOCAL TERMINAL FROM RECEIVING COPY OF
                          OUTPUT TO REMOTE TERMINAL.
 'DISABLE CARRIER ERROR'-CAUSES CONSOLE TO INHIBIT PRINTING OF CARRIER
                          LOST MESSAGE WHEN LOSS OF CARRIER DETECTED.
 'ENABLE LOCAL FLOPPY'   -(AFFECTS PROTOCOL OPERATION ONLY) ON AN ATTEMPT
                          TO OPEN A FILE, THE DIRECTORY OF LOCAL FLOPPY
                          WILL BE SEARCHED FIRST. IF FILE IS NOT FOUND,
                         'REMOTE' FLOPPY'S DIRECTORY IS SEARCHED FOR FILE.
 'DISABLE LOCAL FLOPPY '-(AFFECTS PROTOCOL OPERATION ONLY) ON AN ATTEMPT
                          TO OPEN A FILE, THE FILE IS SEARCHED FOR ON
                          THE 'REMOTE' FLOPPY ONLY.
 'DISABLE REMOTE FLOPPY'-ON AN ATTEMPT TO OPEN A FILE, ONLY THE DIRECTORY
                          OF THE LOCAL FLOPPY WILL BE SEARCHED.  THIS COMMAND
                          AND 'DISABLE LOCAL FLOPPY' ARE MUTUALLY EXCLUSIVE.
 'ENABLE REMOTE FLOPPY' -ALLOWS THE DIRECTORY OF THE 'REMOTE' FLOPPY TO BE
                          SEARCHED ON AN ATTEMPT TO OPEN A FILE.
```

MICRODEBUGGER HELP FILE

```
MICRO-DEBUGGER HELP FILE        REV 2.0, April 13, 1982

DEBUGGER COMMANDS (ALL TERMINATED BY CARRIAGE RETURN)

'E/P <ADDRESS>'                 -EXAMINE PHYSICAL MEMORY

'E/ID <ADDRESS>'                -EXAMINE ID BUS REGISTER

'E <ADDRESS>'                   -EXAMINE WCS LOCATION, DISPLAY ALL FIELDS

'E <ADDRESS> <FIELDNAME-1>,<FIELDNAME-2>,,,,<FIELDNAME-N>
                                EXAMINE WCS LOCATION, DISPLAY ONLY FIELDS
                                THE FIELDS SPECIFIED.
       NOTE: <FIELDNAMES> =     ACF,ACM,ADS,ALU,BEN,BMX,CCK,CID,DK,DT,EAL
                                EBM,FEK,FS,IBC,IEK,UJM,KMX,MCT,MSC,PCK,QK
                                RMX,SCK,SGN,SHF,SI,SMX,SPO,USU,VAK

'E RA <ADDRESS>'                -EXAMINE AN RA REGISTER

'E RC <ADDRESS>'                -EXAMINE AN RC REGISTER

'E <SYMBOLIC-NAME>'              -EXAMINE ONE OF THE SYMBOLICALLY NAMED
                                 REGISTERS
       NOTE: <SYMBOLIC-NAMES> = DR,FER,IBA,LA,LB,LC,Q,RL,SC,SR,UPC

'D/P <ADDRESS> <DATA>' -DEPOSIT <DATA> TO PHYSICAL MEMORY

'D/ID <ADDRESS> <DATA>'         -DEPOSIT <DATA> TO ID BUS REGSITER

'D <ADDRESS> <FIELDNAME-1> <DATA-1>,<FIELDNAME-2> <DATA-2>,.....
                                -DEPOSIT TO WCS LOCATION, PUTTING <DATA-1>
                                 INTO <FIELDNAME-1>, ETC. UNSPECIFIED FIELDS
                                 ARE UNCHANGED.
       NOTE: THE '/Z' QUALIFIER MAY BE USED TO CAUSE ALL UNSPECIFIED
             FIELDS TO BE CLEARED.

'D RA <ADDRESS> <DATA>'         -DEPOSIT <DATA> TO AN RA REGISTER

'D RC <ADDRESS> <DATA>'         -DEPOSIT <DATA> TO AN RC REGISTER

'D <SYMBOLIC-NAME> <DATA>'      -DEPOSIT <DATA> TO ONE OF THE SYMBOLICALLY
                                 NAMED REGISTERS(SEE LIST ABOVE).
       NOTE: DEPOSITS TO THE RLOG STACK(RL) ARE NOT SUPPORTED.

'CONTINUE'                      -RESUME MICRO-INSTRUCTION EXECUTION AS
                                 SPECIFIED BY CONTENTS OF MICRO-PC(UPC)

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

'HALT'                  -HALT THE MICRO-SEQUENCER

'SET SOMM'                      -SET THE 'STOP ON MICRO-MATCH' ENABLE

'CLEAR SOMM'                    -CLEAR THE 'STOP ON MICRO-MATCH' ENABLE

'SET STEP'                      -ENABLE SINGLE MICRO-INSTRUCTION STEP MODE.
                                 START OR CONTINUE WILL ALLOW ONE MICRO-
                                 INSTRUCTION TO EXECUTE, THEN HALT THE
                                 MICRO-SEQUENCER.

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

MICRODEBUGGER HELP FILE (CONT)

```
'OPEN <FILENAME>'                -OPEN SPECIFIED FILE ON FLOPPY DRIVE 0

'OPEN DX1:<FILENAME>'            -OPEN SPECIFIED FILE ON FLOPPY DRIVE 1
       NOTE:   'OPEN' IS USED TO SPECIFY A FILE CONTAINING THE MICRO-CODE
               CURRENTLY LOADED IN THE WCS PORTION OF THE CONTROL STORE.
               (ADDRESSES 1000(16) & UP IN THE CONTROL STORE)
               THIS FILE WILL BE USED FOR ALL EXAMINES OF THE WCS,
               SINCE THE WCS IS NOT DIRECTLY READABLE.

'RETURN'                          -RETURN TO THE CONSOLE PROGRAM.
                                 +-----------------------------------------+
                                 !       DO NOT USE THE RETURN COMMAND     !
                                 !    UNLESS THE CONSOLE FLOPPY IS IN CS1. !
                                 +-----------------------------------------+
                                 ! TO RETURN TO THE CONSOLE PROGRAM, REMOVE !
                                 ! THE WCS DEBUG FLOPPY, INSERT THE CONSOLE !
                                 !     FLOPPY, THEN TYPE 'RETURN <CR>'.    !
                                 +-----------------------------------------+
```

LSI-11 CONSOLE ODT COMMANDS

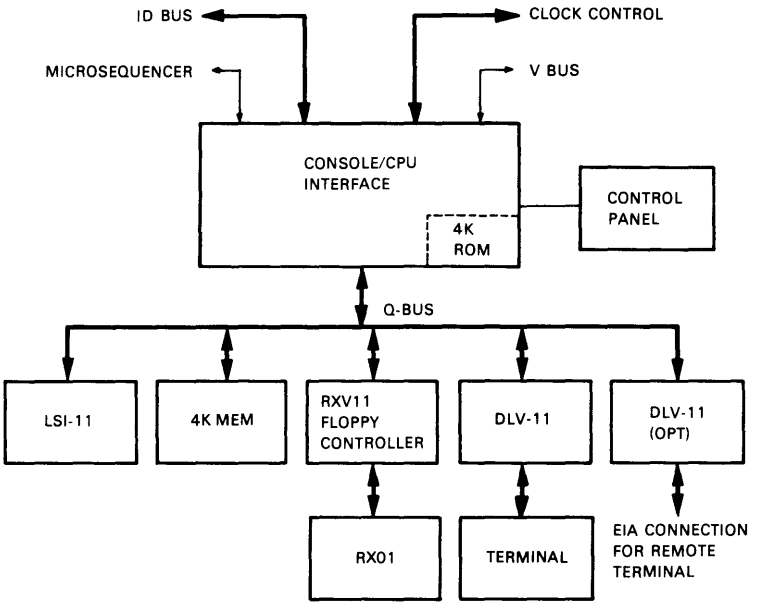
Format	Description
RETURN	Close opened location and accept next command.
LINE FEED	Close current location; open next sequential location.
or]	Open previous location.
or -	Take contents of opened location, index by opened location plus 2, and open that location.
@	Take contents of opened location as an absolute address and open that location.
r/	Open location r.
/	Open last location.
$n or Rn	Open general register n (Ø-7) or S (PS register).
r; G or rG	Go to location r, initialize the bus, and start program.
nL	Execute bootstrap loader using n as device CSR address.
;P or P	Proceed with program execution.
RUBOUT or DELete	Erase previous character. Response is a backslash \ (134) each time RUBOUT is entered.
M	Maintenance. Display of an internal CPU register follows the M command. Only the last digit displayed is significant, indicating how the CPU entered the Halt (ODT) mode, as follows:

Last Digit	Halt Source
Ø or 4	HALT instruction or BHALT L bus signal asserted.
1 or 5	Bus error occurred while getting device interrupt vector.

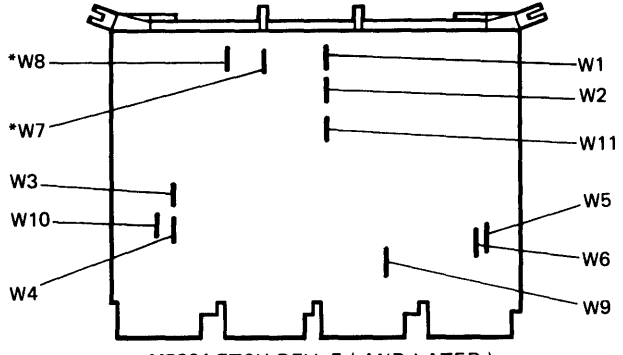
Last Digit	Halt Source
2 or 6	Bus error occurred while doing memory refresh.
3	Double bus error occurred (stack was nonexistent value).
4	Reserved instruction trap occurred (nonexistent micro-PC address occurred on internal CPU bus).
7	A combination of 1, 2, and 4 occurred.

CTRL-SHIFT-S — For manufacturing tests only. Escape this command function by typing NULL and @ (ØØØ and 1ØØ).

CONSOLE SUBSYSTEM CONFIGURATION



JUMPER	IN/OUT	RESULT
W1	–	RESIDENT MEMORY AT A BANK 0
W2	|	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		PRECONFIGURED*
W9	–	ENABLE REPLY FROM RESIDENT MEMORY
W10	|	DISABLE REPLY FROM RESIDENT MEMORY DURING REFRESH
W11	|	ENABLE ON BOARD MEMORY SELECT



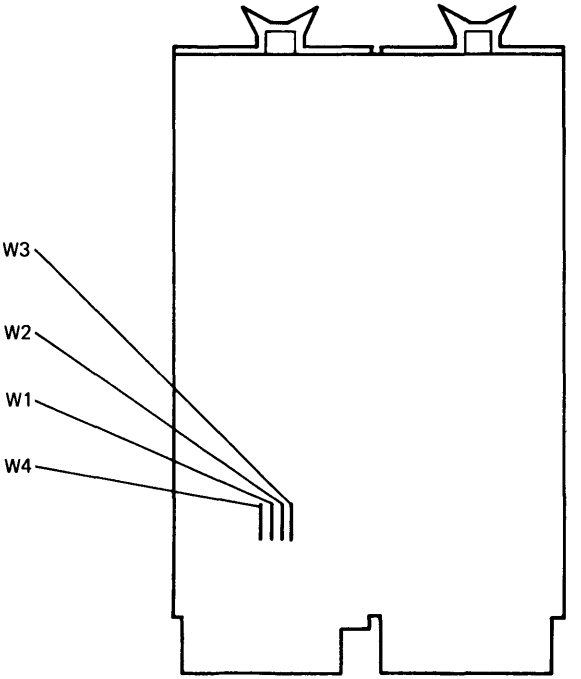
M7264 ETCH REV. E (AND LATER)

*FACTORY CONFIGURED DO NOT CHANGE (W7 & W8)

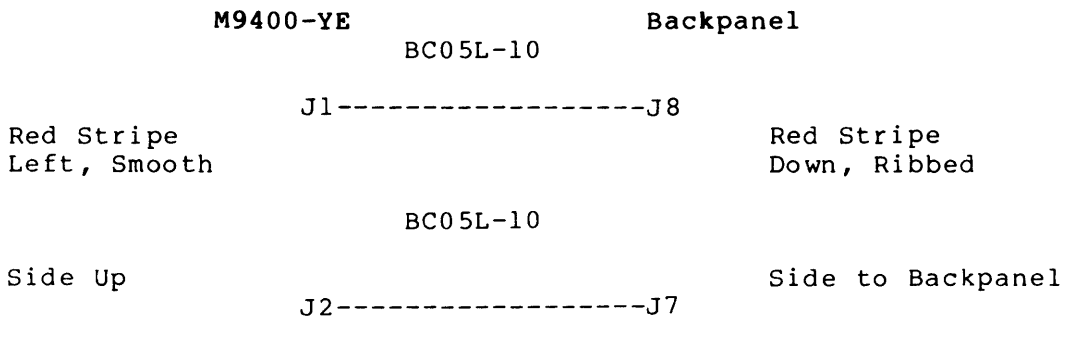
TK-0700

MSV-11B 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 (M7946)

Should be preconfigured for:

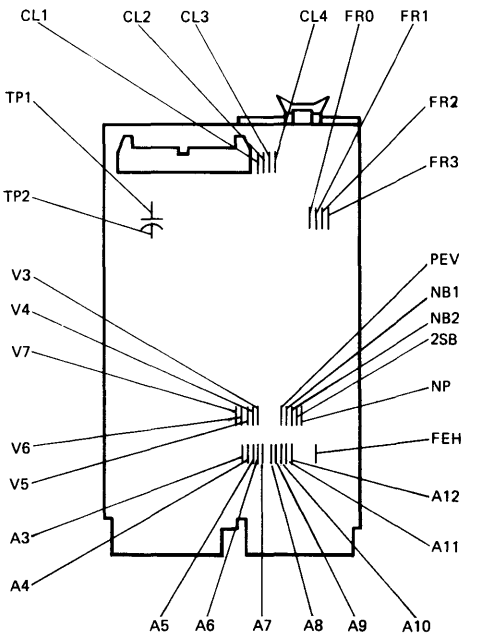
Address: 177170-177172
Vector: 264
Ribbon cable should be installed with
red stripe toward center of module.

DLV11 JUMPER CONFIGURATION

JUMPER	IN/OUT	RESULT
NP	–	NO PARITY
2SB	I	1 STOP BIT
NB2	–	8 DATA BITS
NB1	–	8 DATA BITS
PEV	X	DON'T CARE PARITY EVEN/ODD
FEH	–	NO HALT ON FRAMING ERROR
EIA	–	NO EIA OPERATION
FR3	–	SELECTS 300 BAUD
FR2	–	SELECTS 300 BAUD
FR1	I	SELECTS 300 BAUD
CL4-CL0	I	20 MA ACTIVE XMIT. & RECEIVE

VECTOR JUMPERS SET TO 60-64
V7=1, V6=1, V5=–, V4=1, V3=1

ADDRESS JUMPERS SET TO 177560-177566
A12=–, A11=–, A10=–, A9=–, A8=–,
A7=I, A6=–, A5=–, A4=–, A3=I



DLV11-E JUMPER CONFIGURATION

DLV IIE

R3 R2 R1 R0
I – I –

T3 T2 T1 T0
I – I –

A12 A11 A10 A9 A8 A7 A6 A5 A4 A3
I I – I I I – – – I

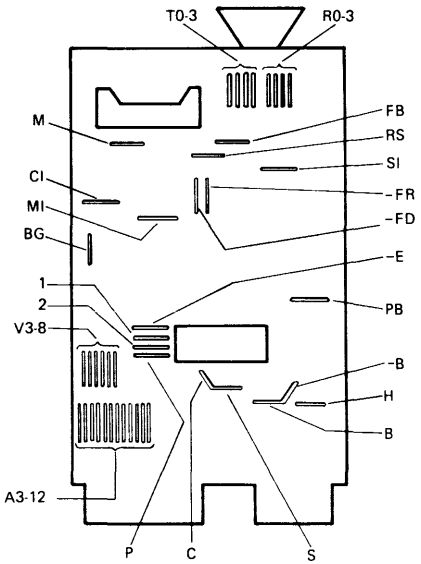
V8 V7 V6 V5 V4 V3
– I I – – I

1 2 P –E PB BG C C1
– – – – – – I I

S S1 H -B B -FR -FD RS
– – – I – I I I

FB M M1
– – –

TK-0726



TK-0718

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 <15:00> 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 respect 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 on the bus for an output transfer from the processor to an I/O slave device. The slave device deskews BDOUT L (pauses) before latching 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	Write/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 (DATO or DATOB) 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 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) or that the device 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 BDIN L and BIAKO L.
BIAKO L and BIAKI L	Interrupt Acknowledge Output and Interrupt Acknowledge Input – The processor asserts this signal in response to an interrupt request (BIRQ L). The processor asserts BIAKO L which is routed via the Q bus to 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 BIAKO L pin and the BIAKI L pin of the lower 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 bank (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.

Q-BUS SIGNAL DESCRIPTION (CONT)

Initialization, Power Fail Signals	
Name	**Description**
BPOK H	Power OK - The power supply asserts this signal when primary power is normal. If BPOK 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 to sustain 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. External interrupts are ignored, but memory refresh interrupts are enabled if W4 on the processor module is removed. When the processor is in the halt state, it executes the ODT microcode, invoking console device (terminal) operation.
BREF L	Memory Refresh - This signal can be asserted by a processor microcode-generated refresh interrupt sequence (when enabled) or by an external device. BREF L forces all dynamic MOS memory units to be activated for each BSYNC L/BDIN L bus transaction.

CONSOLE BOOT/TROUBLESHOOTING FLOW

If the console program does not start and run properly when the VAX-11/780 system is powered up, and a problem in the console subsystem is suspected, proceed as follows.

Action	Response
Turn dc off. Turn ac off. Push HALT/ENABLE switch down (HALT).	
Turn ac on.	
Turn dc on.	DC ON (LED on LSI-11 control panel) 173000 @ (printed on terminal) RUN (light flashes)

If the responses are incorrect, go to the Console DC ON Flowchart.

Examine location 173000 (type 173000/).	173000/000137
Examine location 037776 (type 037776/).	037776/XXXXXX

If the response is not correct, go to the Examine 173000 Flowchart.

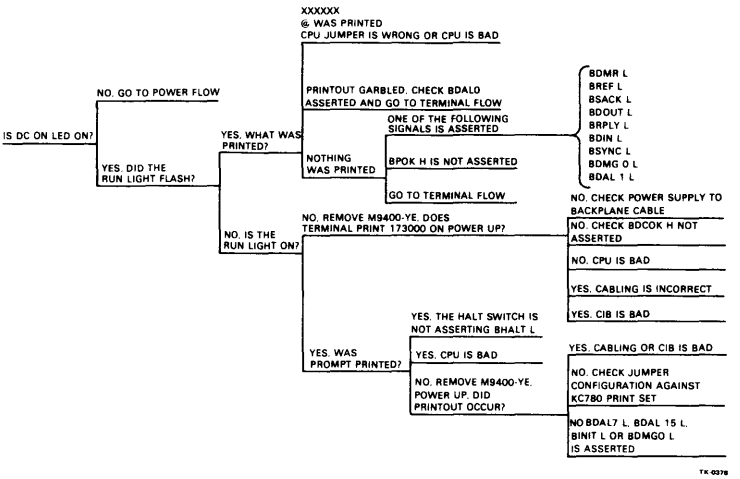
Push HALT/ENABLE switch up (ENABLE).	
Ensure that diskette ZZ-ESZAB is installed properly in the floppy disk drive.	
Type 140200G.	BOOT

This command executes the ROM resident quick check console subsystem diagnostics. On successful completion of these tests, the ROM code boots the console program from the floppy disk. If the boot fails, go to the 140200G Console Boot Failure Flowchart. The program listing for the ROM resident diagnostics (ESKAA.DOC) should be referenced when using this flowchart.

CONSOLE BOOT/TROUBLESHOOTING FLOW (CONT)

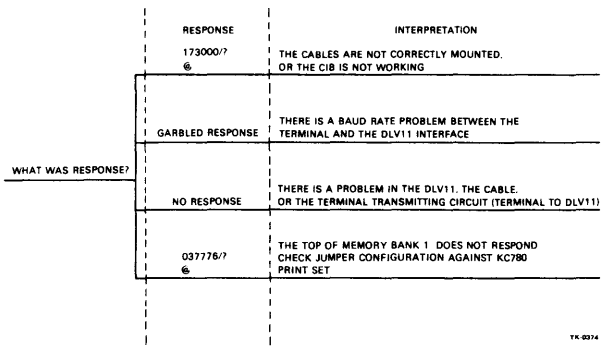
CONSOLE DC ON FLOWCHART

DC ON, RUN LIGHT FLASH,
AND/OR 173000 PRINTOUT DID NOT OCCUR



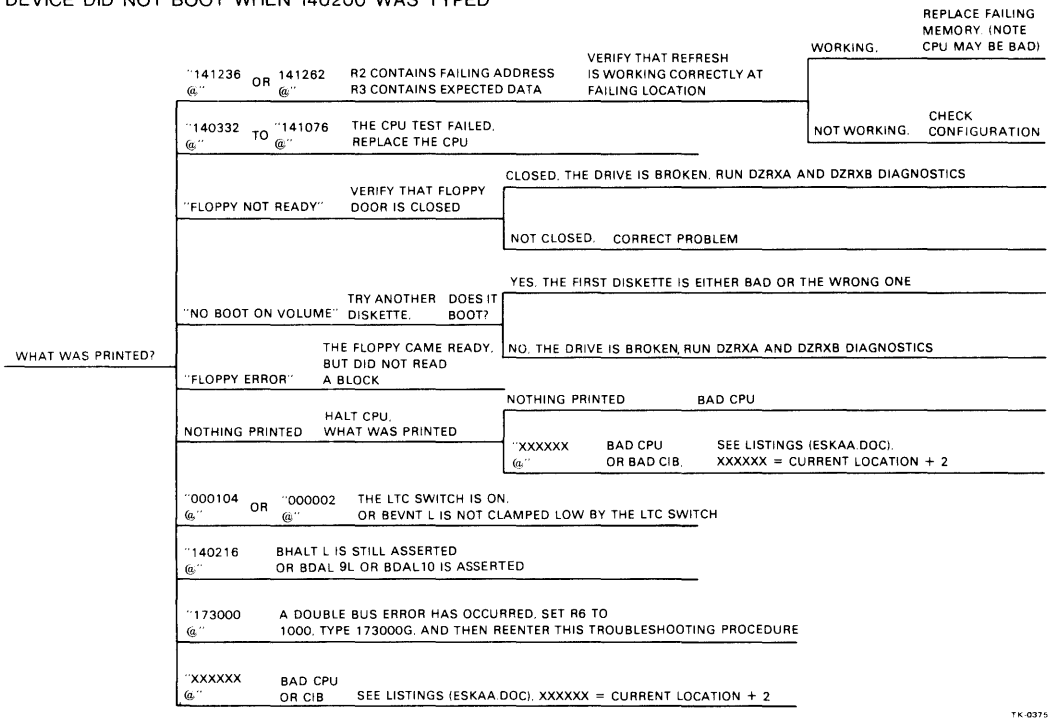
EXAMINE 173000 FLOWCHART

LOCATION 173000 OR LOCATION 037776
DID NOT RESPOND CORRECTLY



140200G CONSOLE BOOT FAILURE FLOWCHART

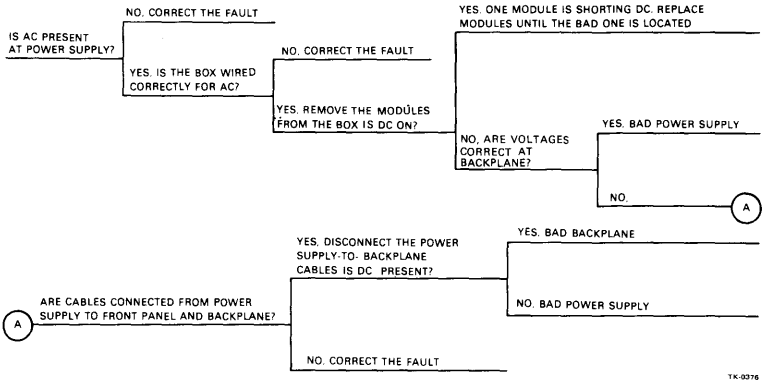
DEVICE DID NOT BOOT WHEN 140200 WAS TYPED



CONSOLE BOOT/TROUBLESHOOTING FLOW (CONT)

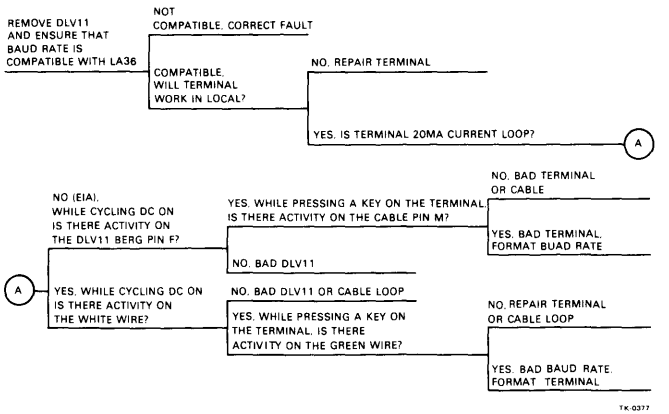
CONSOLE POWER TROUBLESHOOTING FLOWCHART

CONSOLE DC POWER FAILURE



CONSOLE TERMINAL TROUBLESHOOTING FLOWCHART

CONSOLE TERMINAL FAILURE



CONSOLE BOOT/TROUBLESHOOTING FLOW (CONT)

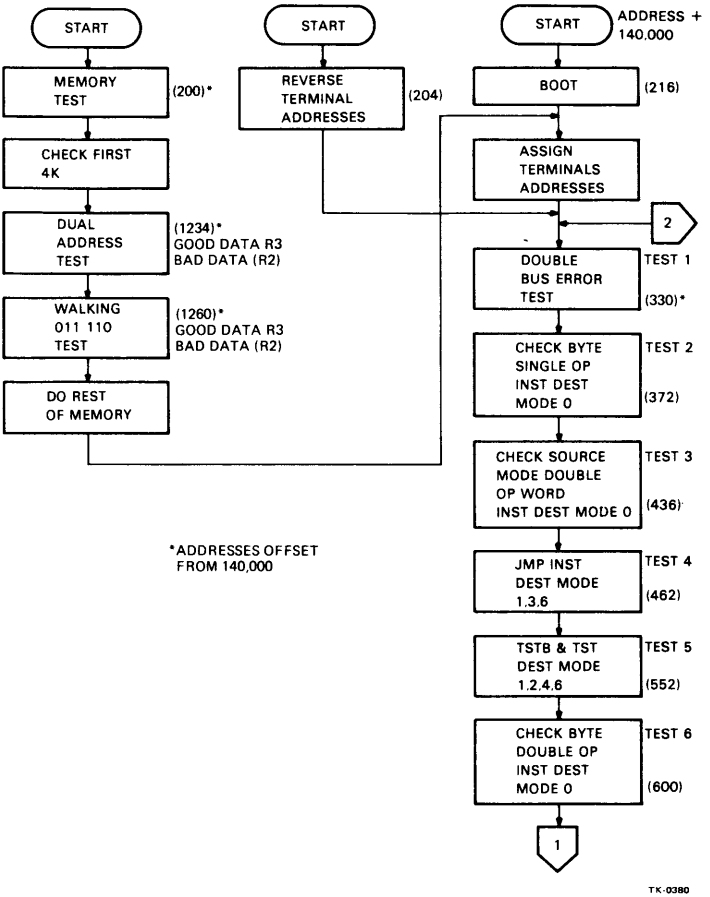
If using the flowcharts fails to help locate a problem, run the RXDP package diagnostics (diskette AS-F824C-MC).

Program Name	Function
VDVA	DLV-11E Test
VDVC	DLV-11F Test
VKAA	LSI-11 CPU Test
VKAB	LSI-11 EIS Instruction Test
VKAE	DLV-11 Test
VKAF	DRV11 Test
ZKMA	Memory Test
ZLAC	LA36 Test
ZRXA	RX11 Disk Exerciser
ZRXB	RX11 Interface Tests

The following figure shows the flow of events in the LSI-11 boot sequence.

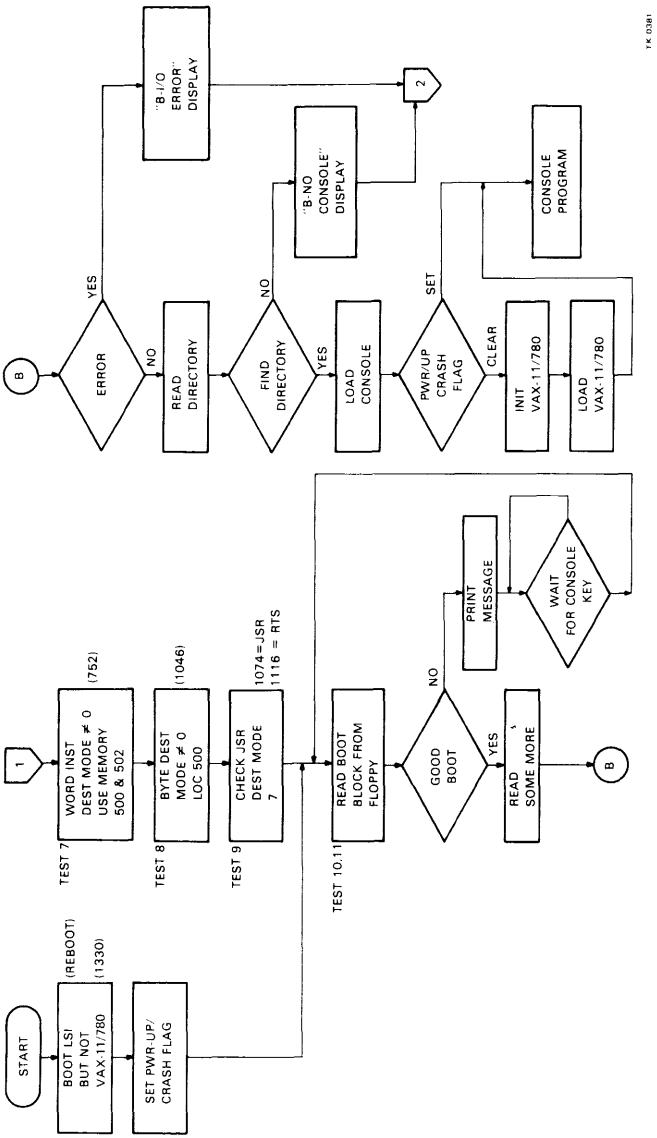
CONSOLE BOOT/TROUBLESHOOTING FLOW (CONT)

LSI-11 BOOT AT 173000 FLOWCHART, PART 1



CONSOLE BOOT/TROUBLESHOOTING FLOW (CONT)

LSI-11 BOOT AT 173000 FLOWCHART, PART 2



CHAPTER 5
INTERNAL REGISTERS

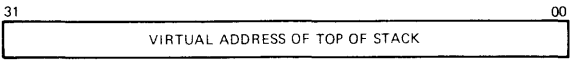
PROCESSOR REGISTER ADDRESSES

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	ASTR	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	MME	Memory management enable	
39	57	TBIA	Translation buffer invalidate all	WO
3A	58	TBIS	Translation buffer invalidate single	WO
3B	59	reserved		
3C	60	MBRK	Microprogram breakpoint	
3D	61	PMR	Performance monitor register	
3E	62	SID	System identification	RO
3F	63	reserved		

PROCESSOR REGISTER BIT CONFIGURATIONS

REG #
DEC. HEX NAME ID#

0 00 KSP 28 **KERNEL STACK POINTER**
1 01 ESP 29 **EXECUTIVE STACK POINTER**
2 02 SSP 2A **SUPERVISOR STACK POINTER**
3 03 USP 2B **USER STACK POINTER**
4 04 ISP 2C **INTERRUPT STACK POINTER**

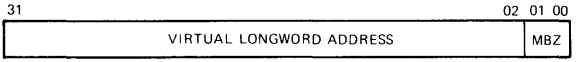


8 08 P0BR 24 **P0 BASE REGISTER**

RESERVED OPERAND FAULT IF VLA < 2**31

10 0A P1BR 25 **P1 BASE REGISTER**

RESERVED OPERAND FAULT IF VLA < 2**31 - 2**21



9 09 P0LR 3C **P0 LENGTH REGISTER**

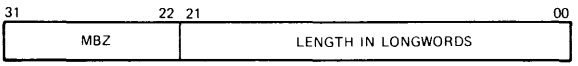
LENGTH OF P0PT IN LONGWORDS

11 0B P1LR 3D **P1 LENGTH REGISTER**

2**21 - LENGTH OF P1PT IN LONGWORDS

13 0D SLR 3E **SYSTEM LENGTH REGISTER**

LENGTH OF SPT IN LONGWORDS
RESERVED OPERAND FAULT IF MBZ ≠0

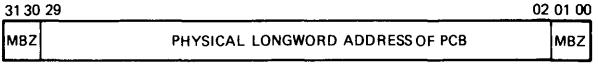


TK-0709

PROCESSOR REGISTER BIT CONFIGURATIONS (CONT)

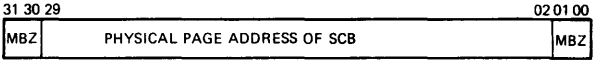
REG. # DEC.	HEX	NAME	ID#	
16	10	PCBB	3A	PROCESS CONTROL BLOCK BASE

RESERVED OPERAND FAULT IF MBZ ≠ 0.

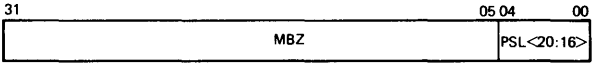


17	11	SCBB	3B	SYSTEM CONTROL BLOCK BASE

RESERVED OPERAND FAULT IF MBZ ≠ 0.

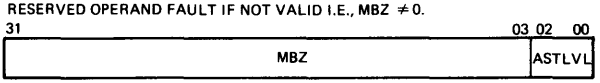


18	12	IPLR	0F	INTERRUPT PRIORITY LEVEL REGISTER



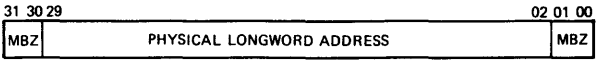
19	13	ASTR	0C	AST LEVEL REGISTER

RESERVED OPERAND FAULT IF NOT VALID I.E., MBZ ≠ 0.



12	0C	SBR	26	SYSTEM BASE REGISTER

RESERVED OPERAND FAULT IF MBZ ≠ 0.

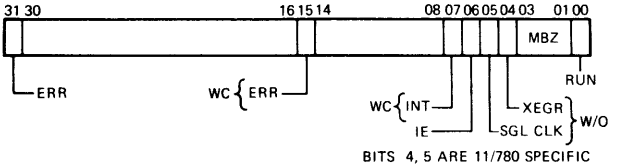


TK-0711

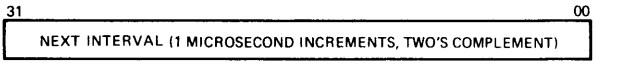
PROCESSOR REGISTER BIT CONFIGURATIONS (CONT)

REG. #

DEC.	HEX	NAME	ID#	
24	18	ICCS	0A	INTERVAL CLOCK CONTROL/STATUS



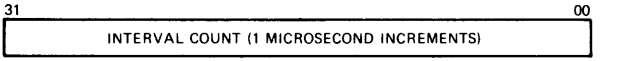
25	19	NICR	09	NEXT INTERVAL COUNT REGISTER



WRITE ONLY

26	1A	ICR	0B	INTERVAL COUNT REGISTER

RESERVED OPERAND FAULT IF WRITE

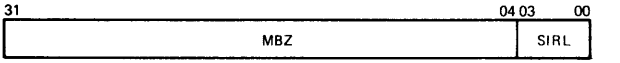


READ ONLY

27	1B	TODR	01	TIME OF DAY REGISTER

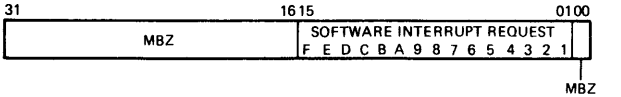
20	14	SIRR		SOFTWARE INTERRUPT REQUEST REGISTER

RESERVED OPERAND FAULT IF READ



WRITE ONLY

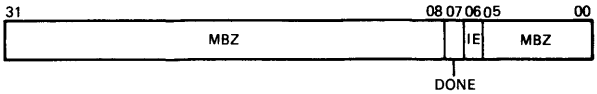
21	15	SISR	0E	SOFTWARE INTERRUPT SUMMARY REGISTER



TK-0710

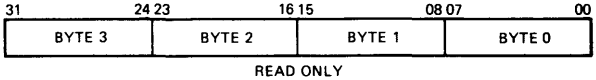
PROCESSOR REGISTER BIT CONFIGURATIONS (CONT)

REG.# DEC	HEX	NAME	ID#	
32	20	RXCS	04	CONSOLE RECEIVE CONTROL/STATUS



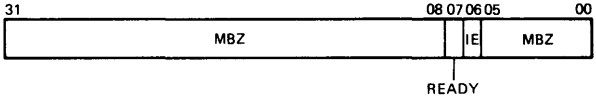
33	21	RXDB	05	CONSOLE RECEIVE DATA BUFFER

RESERVED OPERAND FAULT IF WRITTEN



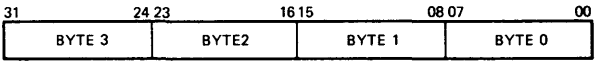
READ ONLY

34	22	TXCS	06	CONSOLE TRANSMIT CONTROL/STATUS



35	23	TXDB	07	CONSOLE TRANSMIT DATA BUFFER

RESERVED OPERAND FAULT IF READ



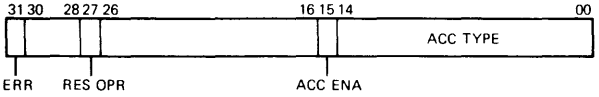
WRITE ONLY

TK-0707

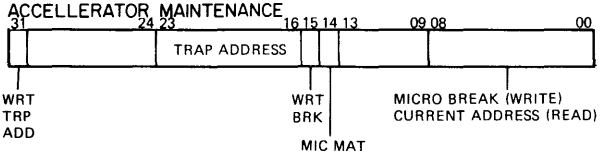
PROCESSOR REGISTER BIT CONFIGURATIONS (CONT)

REG.#
DEC HEX NAME ID#

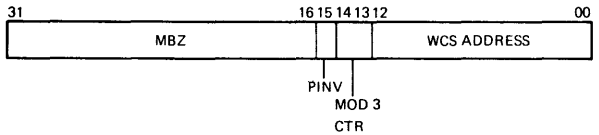
40 28 ACCS 17 ACCELLERATOR CONTROL/STATUS



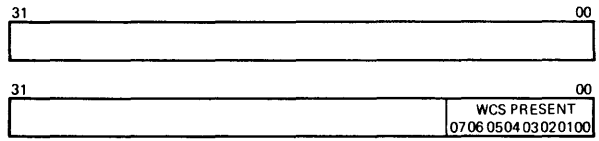
41 29 ACCR 16 ACCELLERATOR MAINTENANCE



44 2C WCSA 22 WRITEABLE CONTROL STORE ADDRESS



45 2D WCSD 23 WRITEABLE CONTROL STORE DATA
WRITE: WCS DATA
READ: WCS PRESENT

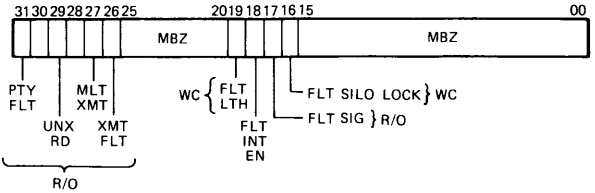


TK-0708

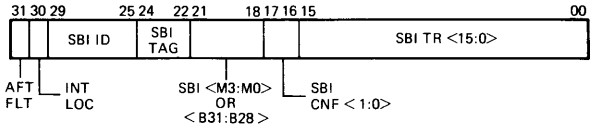
PROCESSOR REGISTER BIT CONFIGURATIONS (CONT)

REG.#
DEC HEX NAME ID#

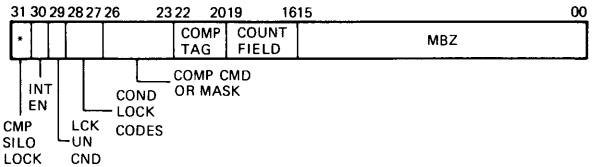
48 30 SBIFS 1B **SBI FAULT/STATUS**



49 31 SBIS 18 **SBI SILO**

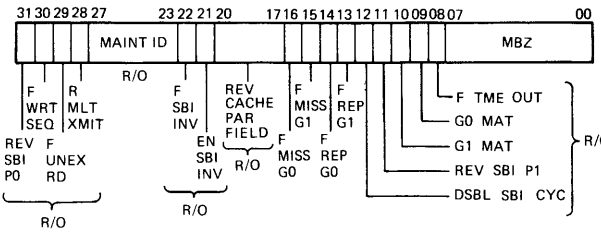


50 32 SBISC 1C **SBI SILO COMPARATOR**



*CLEARED ON ANY WRITE TO SBISC

51 33 SBIMT 1D **SBI MAINTENANCE**

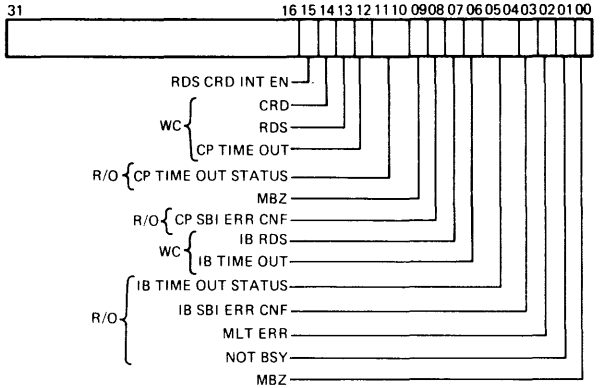


TK-0705

PROCESSOR REGISTER BIT CONFIGURATIONS (CONT)

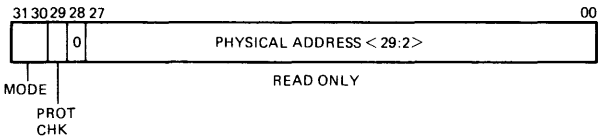
REG.#
DEC HEX NAME ID#

52 34 SBIER 19 **SBI ERROR REGISTER**



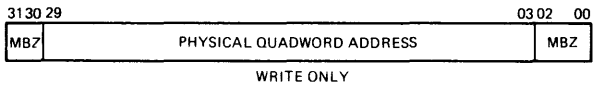
53 35 SBITA 1A **SBI TIMEOUT ADDRESS**

RESERVED OPERAND FAULT IF WRITE



54 36 SBIQC **SBI QUAD CLEAR**

RESERVED OPERAND FAULT IF READ
RESERVED OPREAND FAULT IF MBZ ≠ 0



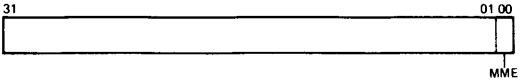
PROCESSOR REGISTER BIT CONFIGURATIONS (CONT)

REG.#
DEC HEX NAME ID#

56 38 MME

MEMORY MANAGEMENT ENABLE

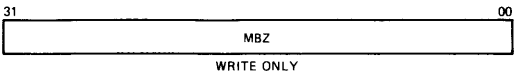
WRITE 1 ALSO CAUSES MICROCODE TO INVALIDATE TB.



57 39 TBIA

TRANSLATION BUFFER INVALIDATE ALL

RESERVED OPERAND FAULT IF READ

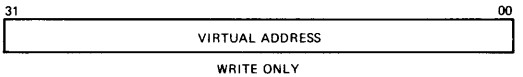


WRITE ONLY

58 3A TBIS

TRANSLATION BUFFER INVALIDATE SINGLE

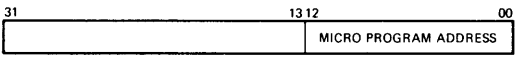
RESERVED OPERAND FAULT IF READ



WRITE ONLY

60 3C MBRK 21

MICROPROGRAM BREAKPOINT



61 3D PMR 0C

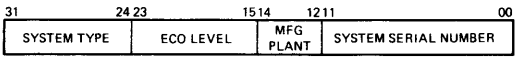
PERFORMANCE MONITOR REGISTER

RESERVED OPERAND FAULT IF >1

62 3E SID 03

SYSTEM IDENTIFICATION

RESERVED OPERAND FAULT IF WRITE



READ ONLY

TK-0704

REG.NAME	REG.ID NO.	INT.REG.NO.	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
IBUF	00		Data Byte 3								Data Byte 2							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			Data Byte 1								Data Byte 0							
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAY.TIME	01	1B TODR	Time Byte 3								Time Byte 2							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			Time Byte 1								Time Byte 0							
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYS.ID	03	3E SID	Type								ECO Level							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			ECO Level	Plant			Serial Number											
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCS	04	20 RXCS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			0	0	0	0	0	0	0	0	Done	Intrpt Enable	0	0	0	0	0	0
RXDB	05	21 RXDB	Data Byte 3								Data Byte 2							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			Data Byte 1								Data Byte 0							
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

REG. NAME	REG. ID NO.	INT. REG. NO.	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
TXCS	06	22 TXCS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			0	0	0	0	0	0	0	0	Ready	Intrpt Enable	0	0	0	0	0	0
TXDB	07	23 TXDB	Data Byte 3								Data Byte 2							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			Data Byte 1								Data Byte 0							
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DQ	08		D(read) Q(write)															
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			D(read) Q(write)															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NXT. PER	09	19 NICR	Next Interval															
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			Next Interval															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLK.CS	0A	18 ICCS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			Error	0	0	0	0	0	0	0	Intrpt Req	Intrpt Enable	Single Clock	Xfer	0	0	0	Run
INTERVAL	0B	1A ICR	Count (microseconds)															
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			Count (microseconds)															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

REG. NAME	REG. ID NO.	INT. REG. NO.	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
CES	0C	13 ASTR 3D PME	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Nested Error
			Control Store Parity Error Summary	2	1	0	EALU N	EALU Z	ALU N	ALU Z	ALU C31	Arithmetic Trap Code			Perf Mon En	AST Level		
VECTOR	0D		0	0	0	0	0	0	Prior Valid	Priority				Number of Ones				
			0	0	0	0	0	0	0	08	07	06	05	Vector 04	03	02	01	00
SIR	0E	15 SISR	0	0	0	0	0	0	0	0	0	0	0	Interrupt Priority Level Active				
			Software Interrupt Register F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
PSL	0F	12 IPL	Compat Mode	Trace Pend	0	0	FPD	Intrpt Stack	Current Mode		Previous Mode		0	Interrupt Priority Level				
			0	0	0	0	0	0	0	0	Decmal Ovrflo	Float Undflo	Intger Ovrflo	T	Condition Codes N	Z	V	C
TBUF	10		Valid	Protection Code				Modify	0	0	0	0	0	Page Frame Number 20	19	18	17	16
			Page Frame Number 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

REG. NAME	REG. ID NO.	INT. REG. NO.	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
TBER0	12		0	0	0	0	0	0	0	0	0	0	0	Replace Both	Force G1	Replace G0	Force G1	TB Misc G0
			FS	ADS	Last Reference MCT 3	MCT 2	MCT 1	MCT 0	1BWCHK	AR	TB Hit G1	G0	0	Force TB Parity Error				Mem Man En
TBER1	13		0	0	0	0	0	0	0	0	0	0	0	1D2	TB Parity Error Bits 1D1	1D0	QD2	QD1
			0D0	1A2	TB Parity Error Bits 1A1	1A0	0A2	0A1	0A0	CP TB PE	0	Last TP Wrp	0	Bad IPA	Miss	IPA PE	Prot E	Auto L
ACC.MN	16		Write Trp Ad	0	0	0	0	0	0	0	Trap Address							
			Write μBreak	Micro Match	0	0	0	0	0	Micro-break (write) Current Address (read)								
ACC.CS	17	28 ACCS	Error	0	0	0	Resrvd Oprand	0	0	0	0	0	0	0	0	0	0	0
			Accel Enable	0	0	0	0	0	0	0	0	0	0	0	Accelerator Type 0	0	0	1
SILO	18	31 SBIS	After Fault	SBI Intlk	4	3	SBI ID 2	1	0	2	SBI TAG 1	0	M3/B31	SBI M2/B30	M1/B29	M0/B28	SBI CNF 1	CNF 0
			15	14	13	12	11	10	9	SBI TR 8	7	6	5	4	3	2	1	0
SBI.ERR	19	34 SBIER	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			RDS Int En	CRD	RDS	CP TO	CP TO ST1	CP TO ST0	0	CP Err CNF	IB RDS	IB TO	IB TO ST1	IB TO ST0	IB Err CNF	Mult Error	Not Busy	0

REG. NAME	REG. ID NO.	INT. REG. NO.	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
TIME.ADR	1A	35 SBITA	Mode 1	Mode 0	Prot Check	0	29	28	27	26	25	Physical Address 24	23	22	21	20	19	18
			17	16	15	14	13	12	11	10	9	Physical Address 8	7	6	5	4	3	2
FAULT	1B	30 SBIFS	Parity Fault	0	Unexp RD	0	Mult Xmit	Xmit Fault	EFP	Spare	0	0	0	0	Fault Latch	Fault Int En	Fault Signal	Fault Lock
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COMP	1C	32 SBISC	Silo Lock	Silo Int En	Lock Uncond	Cond Lock Code			Compare Command or Mask			Compare Tag			Count			
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MAINT	1D	33 SBIMT	Rev P0	Wr Seq Fault	Unexp RD	Mult Xmit	Maintenance ID 4	3	2	1	0	Force SBI Invalid	Enable		Reverse Cache Parity			Force Miss G0
			Force Miss G1	Force Rep G0	Force Rep G1	Disabl SBI	Rev P1	G1 Match	G0 Match	Force T0	0	0	0	0	0	0	0	0
PARITY	1E		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			Any Error	CP Error	G1 B0	G1 B1	G1 B2	G1 B3	Data Parity OK G0 B0	G0 B1	G0 B2	G0 B3	G0 B0	G0 B1	Address Parity OK G0 B2	G1 B0	G1 B1	G1 B2

REG. NAME	REG. ID NO.	INT. REG. NO.	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
USTACK	20		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
						Control Store Address												
						12	11	10	9	8	7	6	5	4	3	2	1	0
UBREAK	21	3C MBRK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			0	0	0	Control Store Address												
						12	11	10	9	8	7	6	5	4	3	2	1	0
WCS ADDR.	22	2C WCSA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			Invert Parity	Mod 3 Counter		Control Store Address												
						12	11	10	9	8	7	6	5	4	3	2	1	0
WCS DATA	23	2D WCSD	Data 31	Data 30	Data 29	Data 28	Data 27	Data 26	Data 25	Data 24	Data 23	Data 22	Data 21	Data 20	Data 19	Data 18	Data 17	Data 16
			Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0

ID-BUS MAP (CONT)

Scratch Pad Locations

Name	Addr	IR No.	Symb		Name	Addr	IR No.	Symb
P0BR	24	08	P0BR		T2	32		
P1BR	25	0A	P1BR		T3	33		
SBR	26	0C	SBR		T4	34		
KSP	28	00	KSP		T5	35		
ESP	29	01	ESP		T6	36		
SSP	2A	02	SSP		T7	37		
USP	2B	03	USP		T8	38		
ISP	2C	04	ISP		T9	39		
FPDA	2D				PCBB	3A	10	PCBB
D. SV	2E				SCBB	3B	11	SCBB
Q. SV	2F				P0LR	3C	09	P0LR
T0	30				P1LR	3D	08	P1LR
T1	31				SLR	3E	0D	SLR
				1457	.BIN			
				1458				

ID-BUS REGISTER BIT CONFIGURATIONS

ID#: 00	NAME: IBUF
Bit Fields	Description
<31:00>	Data in Instruction Buffer Bytes <3:0> Read Only Located on M8223 (IDPL)

ID#: 01	NAME: TIME OF DAY
Bit Fields	Description
<31:00>	32 bit counter 100 Hertz rate Read/Write Located on M8224 (IRCN)

ID#: 03	NAME: SYSTEM ID
Bit Fields	Description
<31:24>	System Type 01=VAX-11/780
<23:15>	ECO Level
<14:12>	Manufacturing Plant
<11:00>	System Serial Number Read Only Selected by jumpers on backpanel Read from M8236 CIBC,D,E

ID#: 04	NAME: RXCS
Bit Fields	Description
<07>	Done Set by console software signifying data available in RXDB Read Only
<06>	Interrupt Enable Allows interrupt when Done set Read/Write Located on M8236 (CIBE)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

ID#: 05	NAME: RXDB
Bit Fields	Description
<31:0>	Data from Console Subsystem Read Only Located on M8236 (CIBC,D,E)
ID#: 06	NAME: TXCS
Bit Fields	Description
<07>	Ready Set by console to indicate ready to receive data Read Only
<06>	Interrupt Enable Allows interrupt when Ready set Read/Write Located on M8236 (CIBE)
ID#: 07	NAME: TXDB
Bit Fields	Description
<31:0>	Data to console subsystem Write Only Located on M8236 (CIBC,D,E)
ID#: 08	NAME: DQ
Bit Fields	Description
<31:00>	Read: D Register Write: Q Register Read/Write Located on: <7:00> M8228 (DCPC) <15:08> M8227 (DDPC) <31:16> M8226 (DEPL,M)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

ID#: 09	NAME: NEXT INTERVAL COUNTER
Bit Fields	Description
<31:00>	Data loaded into interval counter on overflow or XFER bit in CLK CONTL REG Write Only <31:16> M8230 (CEHP) <15:00> M8231 (ICLS)

ID#: 0A	NAME: INTERVAL CLOCK STATUS
Bit Fields	Description
<15>	Error Over run second overflow before first serviced. Read/Write 1 to clear
<07>	Interrupt Request Set when counter overflows Read/Write 1 to clear
<06>	Interrupt Enable Enables interrupt on overflow Read/Write
<05>	Single CLK Advance counter on step Write Only
<04>	XFER Forces next interval to counter Write Only
<00>	RUN Allows counter to increment at 1 microsecond rate Read/Write Located on M8231 (ICLS)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

ID#: 0B	NAME: INTERVAL COUNTER
Bit Fields	Description
<31:00>	32 Bit Up Counter At 1 microsecond rate Read Only <31:16> M8230 (CEHP) <15:00> M8231 (ICLS)

ID#: 0C	NAME: CPU ERROR STATUS (CES)
Bit Fields	Description
<16>	Nested Error Used by Memory Management Microcode Read Only Located on M8230 (CEHP)
<15>	Control Store Parity Error Summary "OR" of Control Store Parity Error Bits Read Only Located on M8231 (ICLS)
<14:12>	Control Store Parity Error Bits <14>=Group 2 <13>=Group 1 <12>=Group 0 Read Only Located on M8231 (ICLS)
<11>	E ALU N
<10>	E ALU Z
<09>	ALU N
<08>	ALU Z
<07>	ALU C31 Read/Write Located on M8231 (ICLS)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<Ø6:Ø4>	Arithmetic Trap Code 7=Decimal Divide by Ø 6=Decimal Overflow 5=Float Underflow 4=Float Divide by Ø 3=Float Overflow 2=Integer divide by Ø 1=Integer Overflow Ø=No Trap Pending Read/Write Located on M8231 (ICLS)
<Ø3>	Performance Monitor Enable Loaded or read by microcode Read/Write Located on M8231 (ICLS)
<Ø2:Ø1>	AST Level Used to deliver AST SIR during RET Read/Write Located on M8231 (ICLS)

ID#: ØD	NAME: VECTOR
Bit Fields	Description
<25>	Prior Valid Indicates at least one bit was set in last priority field Read Only Located on M8230 (CEHP)
<24:21>	Priority Priority encoded value of bits <31:16> of bit mask last written into vector register Read Only Located on M8230 (CEHP)
<20:16>	Number Of Ones Number of ones last written into vector register Read Only Located on M8230 (CEHP)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<08:00>	Vector Hardware generated vector Read Only Located on M8231 (CEHJ)
ID#: 0E	NAME: SOFTWARE INTERRUPT REGISTER
Bit Fields	Description
<20:16>	Interrupt Priority Level Pending Level of highest interrupt active at last interrupt strobe time Read Only Located on M8230 (ICLS)
<15:01>	Software Interrupt Register Pending software interrupt flags Read/Write Located on M8231 (ICLS)
ID#: 0F	NAME: PROCESSOR STATUS LONGWORD
Bit Fields	Description
<31>	Compatibility Mode CPU executing PDP-11 mode instructions Read/Write Located on M8230 (CEHP)
<30>	Trace Pending At end of an instruction and if trace pending equal a trace trap is initiated Read/Write Located on M8230 (CEHP)
<27>	First Part Done Microcode sets this bit at defined points within certain instructions, stating that instruction may be restarted from that point if an interrupt of instruction occurs. Read/Write Located on M8230 (CEHP)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<26>	Interrupt Stack Indicates CP operating on interrupt stack Read/Write Located on M8230 (CEHP)
<25:24>	Current Mode Current operating mode of software 3=USER 2=SUPERVISOR 1=EXECUTIVE 0=KERNEL Read/Write Located on M8230 (CEHP)
<23:22>	Previous Mode Previous operating mode (before change mode instruction) 3=User 2=Supervisor 1=Executive 0=Kernel Read/Write Located on M8230 (CEHP)
<20:16>	Interrupt Priority Level Current interrupt priority level of CPU Read/Write Located on M8230
<07>	Enable decimal overflow exceptions
<06>	Enable floating underflow exceptions
<05>	Enable integer overflow exceptions Read/Write Located on M8231 (ICLS)
<04>	T bit Results in setting Trace Pending
<03>	N bit
<02>	Z bit
<01>	V bit

<00> C bit

Read/Write
Located on M8231 (ICLS)

ID#: 10 NAME: TRANSLATION BUFFER DATA REGISTER

Bit Fields Description

<31> Valid

Allows TB hits with VA<13:9> and 31 used as index and address<30:14> equals VA MUX<30:14>

Write Only
Located on M8220 (CAMV)

<30:27> Protection Code

Define Protection of Address

	Kernel	Exec	Super	User
0000	*	*	*	*
0001	Unpredictable			
0010	R/W	*	*	*
0011	R0	*	*	*
0100	R/W	R/W	R/W	R/W
0101	R/W	R/W	*	*
0110	R/W	R0	*	*
0111	R0	R0	*	*
1000	R/W	R/W	R/W	*
1001	R/W	R/W	R0	*
1010	R/W	R0	R0	*
1011	R0	R0	R0	*
1100	R/W	R/W	R/W	R0
1101	R/W	R/W	R0	R0
1110	R/W	R0	R0	R0
1111	R0	R0	R0	R0

* No access
R/W Read/Write
R0 Read

Write Only
Located on M8220 (CAMV)

<26> Modify

Notes a modified page

Write Only
Located on M8220 (CAMV)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<20:0>	Page Frame Number When translation occurs these bits become page numbers, i.e., PA<29:09> Write Only Located on M8222 (TBME)
ID#: 12	NAME: T BUFF REG 0
Bit Fields	Description
<20:18>	Force Replace Directs TB writes to defined groups 20=Write Both 19=Force Replace Group 1 18=Force Replace Group 0 Read/Write Located on M8222 (TBME)
<17:16>	Force Miss Force TB miss on defined group 17=Group 1 16=Group 0 Read/Write Located on M8222 (TBME)
<15:08>	Last Reference Data on last non-nop memory reference <15> Status of uFS bit <14> Status of uADS bit <13:10> Status of uMCT field <09> 1 means IB WCHK existed on an IB reference <08> 1 means reference delayed one cycle by IB auto reload Read Only Located on M8222 (TBME)
<07:06>	TB Hit Indicate which group was a TB hit 7=Group 1 6=Group 0 Read Only Located on M8222 (TBME)

<04:01>	Force TB Parity Error Allows bad parity to be generated 0 No errors 1 No errors 2 Group 0 Data Byte 0 3 " 0 " " 1 4 " 0 " " 2 5 " 1 " " 0 6 " 1 " " 1 7 " 1 " " 2 8 " 0 Address Byte 0 9 " 0 " " 1 A " 0 " " 2 B " 1 " " 0 C " 1 " " 1 D " 1 " " 2 E No errors F No errors Read/Write Located on M8222 (TBME)
<00>	MME Enable Memory Management Read/Write Located on M8222 (TBME)
ID#: 13 Bit Fields	NAME: TBUFF REG 1 Description
<20:09>	TB Parity Error Status 20=1 Group 1 Data Byte 2 19=1 " 1 " " 1 18=1 " 1 " " 0 17=1 " 0 " " 2 16=1 " 0 " " 1 15=1 " 0 " " 0 14=1 " 1 Address Byte 2 13=1 " 1 " " 1 12=1 " 1 " " 0 11=1 " 0 " " 2 10=1 " 0 " " 1 9=1 " 0 " " 0 Read/Any Write Clears Located on M8222 (TBME)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<Ø8>	CP TB Parity Error Indicate TB microtrap has been requested Read/Any Write Clears Located on M8222 (TBME)
<Ø6>	Last TB Write Pulse Indicates which TB group was last written Ø=Group Ø 1=Group 1 Both - Unpredictable Read Only Located on M8222 (TBME)
<Ø4>	Bad IPA Contents of IPA are not meaningful Read Only Located on M8222 (TBME)
<Ø3:Ø0>	IPA Info Status of last load from IPA 3=1 TB miss on load 2=1 TB parity error 1=1 Protection violation or miss Ø=1 Automatic hardware initiated load Read Only Located on M8222 (TBME)

ID#: 16	NAME: ACCELERATOR MAINTENANCE
Bit Fields	Description
<31>	Write Trap Address When set clocks trap address register Write Only Located on M8286 (FMHR)
<23:16>	Trap Address Use to form ROM address on ACC trap Read/Write Located on M8286 (FMHR)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<15>	Write Micro Match Setting clocks micro match register from bits <8:0> of this register Write Only Located on M8287 (FMLP)
<14>	Micro Match Indicates a micro match has occured Read Only Located on M8287 (FMLP)
<08:00>	Micro Break/Current Address Writes micro break register Reads current micro program counter Read/Write Located on M8287 (FMLP)

ID#: 17	NAME: ACCELERATOR CONTROL STATUS
Bit Fields	Description
<31>	Error Read/Any write to this register will clear Located on M8286 (FMHR)
<27>	Reserved Operand Minus zero error Read Only Located on M8286 (FMHR)
<15>	Accelerator Enable 1=Enable Accelerator 0=Disable Accelerator Read/Write Located on M8287 (FMLP)
<03:00>	Accelerator type 01=FPA Read Only Located on M8287 (FMLP)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

ID#: 18	NAME: SILO
Bit Fields	Description
	16 location SILO used to store SBI activity
<31>	After Fault
	First entry after fault cleared
	Read Only Located on M8219 (SBHJ)
<30>	SBI Interlock
	Read Only Located on M8219 (SBHJ)
<29:25>	SBI ID<4:0>
	Read Only Located on M8219 (SBHJ)
<24:22>	SBI TAG<2:0>
	Read Only Located on M8219 (SBHJ)
<21:18>	SBI MASK<3:0> or SBI<B31:B28>
	Silo written with SBI<B31:B28> when SBI TAG equals command address. Otherwise SBI <M3:M0> are written
	Read Only Located on M8219 (SBHJ)
<17:16>	SBI CNF<1:0>
	Read Only Located on M8219 (SBHJ)
<15:00>	SBI TR<15:00>
	Read Only Located on M8237 (TRSF)

ID#: 19	NAME: SBI ERROR REGISTER
Bit Fields	Description
<15>	RDS Interrupt Enable
	Enable interrupt for RDS errors
	Read/Write Located on M8218 (SBLH)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<14> CRD

Received corrected read data from memory

Read/Write 1 to clear
Located on M8218 (SBLH)

<13> RDS

Received read data substitute from memory

Read/Write 1 to clear
Located on M8218 (SBLH)

<12:10> CP Timeout Status

12=1 Timeout for CP requested cycle

11	10	
0	0	No device response
0	1	Device busy
1	0	Waiting for read data
1	1	Impossible code

12 - Read/Write 1 to clear
Also clears bits<11:10>, 08, 02
<11:10> Read Only
Located on M8218 (SBLH)

<8> CP SBI Error Confirmation

Set when CP requested cycle receives error confirmation to command address transfer

Read Only
Write 1 to bit 12 to clear
Located on M8218 (SBLH)

<07> IB RDS

Read data substitute for IB data

Read/Write 1 to clear
Located on M8218 (SBLF)

<06:04> IB Timeout Status

06=1 Timeout for IB 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 bits<5:3>
05:04 - Read Only
Located on M8218 (SBLE)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<Ø3>	IB SBI Error Confirmation Set when IB requested cycle receives error confirmation Read Only Write 1 to bit 6 to clear Located on M8218 (SBLF)
<Ø2>	Multiple CP Error Set with pending CP timeout or CP SBI error confirmation not serviced Read Only Write 1 to bit 12 to clear Located on M8218 (SBLF)
<Ø1>	SBI Not Busy Read Only Located on M8218 (SBLF)

ID#: 1A	NAME: TIMEOUT ADDRESS
Bit Fields	Description
	Latches physical address on SBI timeout; will not latch for IB data timeouts Read Only Latched until CP timeout Error bit (SBI ERR REG bit 12)=1
<31:3Ø>	Mode 31 3Ø Ø Ø Kernel Ø 1 Executive 1 Ø Supervisor 1 1 User Located on M8219 (SBHJ)
<29>	Protection Check Equal Ø for references not subject to hardware protection check Located on M8219 (SBHJ)
<27:ØØ>	Physical Address <27:ØØ>=PA<29:Ø2> Located on <27:16> (SBHH,J) <16:ØØ> (SBLF,H)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

ID#: 1B	NAME: SBI FAULT STATUS REGISTER
Bit Fields	Description
<31>	Parity Fault SBI Parity Fault Read Only Located on M8219 (SBHJ)
<29>	Unexpected Read Data Fault Read Only Located on M8219 (SBHJ)
<27>	Multiple Transmitter Fault Read Only Located on M8219 (SBHJ)
<26>	Transmitter During Fault Read Only Located on M8219 (SBHJ)
<25>	Error First Pass Set by microcode first time through fault handling code; used to note double errors Read/Write Located on M8219 (SBHJ)
<24>	Spare Read/Write Located on M8219 (SBHJ)
<19>	Fault Latch Set from SBI fault Read/Write 1 to clear Located on M8219 (SBHH)
<18>	Fault Interrupt Enable Interrupt on SBI fault enable Read/Write Located on M8219 (SBHH)
<17>	SBI Fault Signal Read Only Located on M8219 (SBHH)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<16>	Fault Silo Lock Indicates SBI Silo locked from SBI fault Read Only Write 1 to bit 19 to clear Located on M8219 (SBHH)
ID#: 1C	NAME: SILO COMPARATOR
Bit Fields	Description
	Allows lock of silo on predetermined data other than fault
<31>	Comp Silo Lock A. Lock unconditional (see bit 29) Locks when counter (bits 19:16) equals F B. Conditional lock Lock when certain conditions exist. Comparator looks at SBI. When match, compare signal is generated which allows counter to increment. When counter equals F, silo will lock Unlock by writing number equals F into counter Read Only Clear by writing number not equal F to counter Located on M8219 (SBHJ)
<30>	Silo Lock Interrupt Enable Read/Write Located on M8219 (SBHJ)
<29>	Lock Unconditional Enables silo lock when counter equals F Read/Write Located on M8219 (SBHJ)
<28:27>	Conditional Lock Codes 28 27 0 0 No compare 0 1 ID only 1 0 ID TAG 1 1 ID TAG, command function or mask Read/Write Located on M8219 (SBHJ)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<26:23>	Compare Command or Mask<3:Ø> Read/Write Located on M8219 (SBHJ) (SBHH)
<22:2Ø>	Compare Tag<2:Ø> Read/Write Located on M8219 (SBHH)
<19:16>	Count Field<3:Ø> When equals F, allows silo lock Read/Write Located on M8219 (SBHH)
ID#: 1D	NAME: MAINTENANCE REGISTER
Bit Fields	Description
<31>	Force PØ Reversal on SBI Read/Write Located on M8219 (SBHJ)
<3Ø>	Force Write Sequence Fault Read/Write Located on M8219 (SBHJ)
<29>	Force Unexpected Read Data Fault Causes transmit of SBI TAG=Ø, Maintenance ID, Undefined Data, good parity for unexpected read data in a selected nexus Read/Write Located on M8219 (SBHJ)
<28>	Force Multiple Transmitter Fault
<27:23>	Maintenance ID<4:Ø> Used to force unexpected read data faults Read/Write Located on M8219 (SBHJ) (SBHH)
<22>	Force SBI Invalidate Forces writes done by CPU on SBI to become cache invalidates Read/Write Located on M8219 (SBHH)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<21>

Enable SBI Invalidate

Allows SBI writes to invalidate cache
Must be =1 for normal operation

Read/Write
Located on M8219 (SBHH)

<20:17>

Reverse Cache Parity

20	19	18	17	Reverse Parity
Ø	Ø	Ø	Ø	No P
Ø	Ø	Ø	1	Group 1 Byte A Address
Ø	Ø	1	Ø	Group 1 Byte B Address
Ø	Ø	1	1	Group 1 Byte C Address
Ø	1	Ø	Ø	Group Ø Byte A Address
Ø	1	Ø	1	Group Ø Byte B Address
Ø	1	1	Ø	Group Ø Byte C Address
Ø	1	1	1	Unused
1	Ø	Ø	Ø	Group 1 Byte 3 Data
1	Ø	Ø	1	Group 1 Byte 2 Data
1	Ø	1	Ø	Group 1 Byte 1 Data
1	Ø	1	1	Group 1 Byte Ø Data
1	1	Ø	Ø	Group Ø Byte 3 Data
1	1	Ø	1	Group Ø Byte 2 Data
1	1	1	Ø	Group Ø Byte 1 Data
1	1	1	1	Group Ø Byte Ø Data

Read/Write
Located on M8219 (SBHH)

<16:15>

Force Cache Miss

16	15	
Ø	Ø	No miss forced
Ø	1	Force miss Group 1
1	Ø	Force miss Group Ø
1	1	Force miss Groups Ø,1

Read/Write
Located on <16> M8219 (SBHH)
<15> M8218 (SBLH)

<14:13>

Cache Replacement

14	13	
Ø	Ø	Random
Ø	1	Group 1 always
1	Ø	Group Ø always
1	1	Undefined

Read/Write
Located on M8218 (SBLH)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<12>	Disable SBI When set, no SBI cycles will be started Read/Write Located on M8218 (SBLH)
<11>	Force Pl Reversal on SBI Read/Write Located on M8218 (SBLH)
<10:09>	Cache Match 10=1 Group 1 cache match 09=1 Group 0 cache match Read Only Located on M8218 (SBLH)
<08>	Force Timeout Forces read timeouts Read/Write Located on M8218 (SBLH)

ID#: 1E	NAME: CACHE PARITY ERROR REGISTER
Bit Fields	Description
<15:14>	Error Bit Bit 15 14 0 1 No error 1 0 IB read reference caused error 1 1 CP read reference caused error 15 Read/write 1 clears entire register located on M8218 (SBLH) 14 Read only. Located on M8218 (SBLH)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

<13:06>	Data Parity O.K. If set, parity O.K., bit 15 must be set for meaningful information 13 Parity OK CDM Group 1 Byte Ø 12 " " " " 1 " 1 11 " " " " 1 " 2 1Ø " " " " 1 " 3 9 " " " " Ø " Ø 8 " " " " Ø " 1 7 " " " " Ø " 2 6 " " " " Ø " 3 Read Only Located on M8218 <13:8> (SBLH) <7:6> (SBLF)
<Ø5:ØØ>	Address Parity O.K. If set, parity O.K., bit 15 must be set for meaningful information 5 Parity OK CAM Group Ø Byte Ø 4 " " " " Ø " 1 3 " " " " Ø " 2 2 " " " " 1 " Ø 1 " " " " 1 " 1 Ø " " " " 1 " 2 Read Only Located on M8218 (SBLF)
ID#: 2Ø	NAME: USTACK
Bit Fields	Description
<15:ØØ>	Reading pops top address from micro stack Writing pushes address on micro stack <15:ØØ> = Control Store Address<15:ØØ> Read/Write Located on M8235 (USCD)
ID#: 21	NAME: UBREAK
Bit Fields	Description
<12:ØØ>	Data used to compare micro PC for scope sync or stopping system clock when SOMM set Read/Write Located on M8235 (USCD)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

ID#: 22	NAME: WCS ADDRESS
Bit Fields	Description
<15>	Invert Parity When set inverts WCS parity Read/Write Located on M8235 (USCD)
<14:13>	Modulo 3 Counter Counter used to point to which 32 bit quantity of WCS is to be written Read/Write Located on M8235 (USCD)
<12:00>	Control Store Address Use to address WCS for writing Read/Write Located on M8235 (USCD)

ID#: 23	NAME: WCS DATA
Bit Fields	Description
<31:00>	Data Used to write data into WCS
<07:00>	Number of WCS Boards Present 0=1 0-1K Present 1= 1-2K " 2= 2-3K " 3= 3-4K " 4= 4-5K " 5= 5-6K " 6= 6-7K " 7= 7-8K " <31:8>Write Only <7:0>Read/Write Located on M8233 (WCSB)

ID-BUS REGISTER BIT CONFIGURATIONS (CONT)

ID#	Name	
24	PØBR	All Registers <31:ØØ>
25	PlBR	
26	SBR	<31:24> M8230 CEHN
		<23:16> M8230 CEHM
28	KSP	<15:Ø8> M8231 ICLR
29	ESP	<Ø7:ØØ> M8231 ICLP
2A	SSP	
2B	USP	
2C	ISP	ID registers 24 through 2F are
2D	FPDA	stored in A temps on CEHK, ICLL
2E	D.SY	
2F	Q.SY	
3Ø	TØ	
31	Tl	ID registers 3Ø through 3E are
32	T2	stored in B temps on CEHK, ICLL
33	T3	
34	T4	
35	T5	
36	T6	All registers are Read/Write
37	T7	
38	T8	
39	T9	
3A	PCBB	
3B	SCBB	
3C	PØLR	
3D	PlLR	
3E	SLR	

SILO REGISTER INTERPRETATION

The SBI silo is a read only register file that provides temporary storage of various SBI signals for the last 16 SBI cycles.

The assertion of fault by any nexus locks the silo, sets fault silo lock in the fault register, and makes the data available through the silo register. The silo may also be locked through the use of the SBI comparator register, but comp silo lock will set in the comparator register rather than fault silo lock.

Examining the silo register when the silo is not locked will result in all zeros being returned.

Following is a breakdown of the silo register and a description of the various fields.

AFTER Fault (31)
: Set for first entry after fault clears.

SBI Interlock (INTLK 30)
: The interlock line is asserted by the commander nexus when issuing the interlock read and then by memory when asserting the ACK confirmation.

Identifier Field (ID 29:25)
: Identifies the logical source or destination of information, depending on the TAG type.

TAG Type	ID
Command address	Source
Write data	Source
Interrupt summary read	Source
Read data	Destination

ID code corresponds to the TR line at which the device operates.

ID Code	Device	TR
00001	Memory Adapter 1	1
00011	UNIBUS Adapter 1	3
01000	MASSBUS Adapter 1	8
01001	MASSBUS Adapter 2	9
10000	Processor	16

TAG Field (TAG 24:22)
: Defines the transmit or receive information types.

TAG Type	
000	Read Data
011	Command Address
101	Write Data
110	Interrupt Summary Read

SILO REGISTER INTERPRETATION (CONT)

Function Field (F 21:18)

Used with command address TAG to specify the command type. Silo bits 21:18 are written with function bits B31:B28 when the SBI specified a command address, otherwise SBI mask bits are written here.

Function Code	Function Definition
0000	Reserved
0001	Read Masked
0010	Write Masked
0011	Reserved
0100	Interlock Read Masked
0101	Reserved
0110	Reserved
0111	Interlock Write Masked
1000	Extended Read
1001	Reserved
1010	Reserved
1011	Extended Write Maskked
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

Mask Field (M 21:18)

Primary function: Specify particular data bytes of an addressed location for an operation.

Mask	Byte
0001	0
0010	1
0100	2
1000	3

Second function: Specify particular read data types during a read.

Mask	Data Type
0000	Read Data
0001	Corrected Read Data
0010	Read Data Substitute

Confirmation Code (C 17:16)

00	No response	(N/R) or unasserted
01	Acknowledge	(ACK)
11	Error	(ERR)
10	Busy	

NOTE: No response is normal when there is no activity on SBI.

Arbitration Field (TR Lines 15:00)

Indicates the TR devices that are requesting access to and control of the SBI.

Example of Interpreting a Deposit Byte:

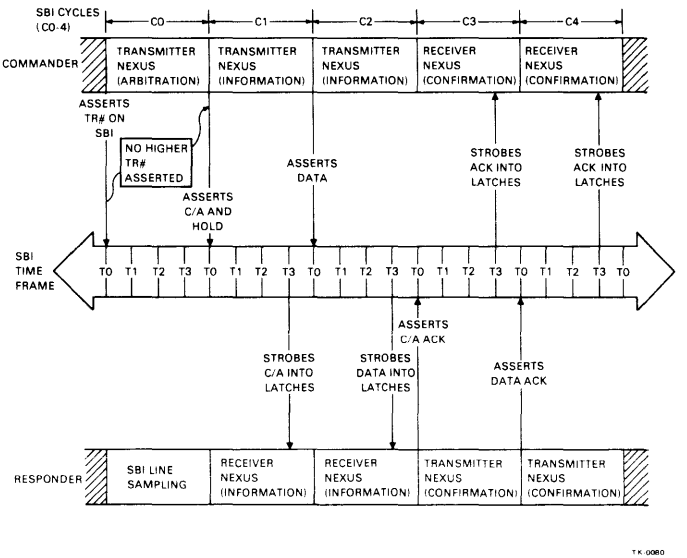
```
D/B 500 AA
```

The following would appear in the silo register if it was locked.

```
R E/ID 18
```

Cycles	Silo	Bit Breakdown
C1	ID00000018 20C80001	ID = CPU, TAG = C/A, FUN = WRITE MASK, TR = HOLD
C2	ID00000018 21440000	ID = CPU, TAG = WRITE DATA, MASK = BYTE 0
C3	ID00000018 00010000	CNF = ACK
C4	ID00000018 00010000	CNF = ACK

These cycles correspond to the following figure.



MICROCODE MACHINE CHECK ERROR LOGOUT

At any machine check, the error handling microcode attempts to log out 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 VAX-11/780 specific, of course, and does not apply to other members of the family.

Data	Memory Location	ID Location	Notes
Byte Count	(SP)	None	40(dec) = 28 (hex)
Summary Parameter	(SP)+4	10 (30)	See below
CPU Error Status	(SP)+8	T1 (31)	See CES register format
Trapped UPC	(SP)+12	T2 (32)	Microcode error location
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 Address	(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 nonzero 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 indentifies the type of machine check:

00 - CP Read Timeout or Error Confirmation Fault
02 - CP Translation Buffer Parity Error Fault
03 - CP Cache Parity Error Fault
05 - CP Read Data Substitute Fault
0A - IB Translation Buffer Parity Error Fault
0C - IB Read Data Substitute Fault
0D - IB Read Timeout or Error Confirmation Fault
0F - IB Cache Parity Error Fault
F1 - Control Store Parity Error Abort
F2 - CP Translation Buffer Parity Error Abort
F3 - CP Cache Parity Error Abort
F0 - CP Read Timeout or Error Confirmation Abort
F5 - CP Read Data Substitute Abort
F6 - Microcode "not supposed to get here" abort

"IB" 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.

DOUBLE ERROR HALT

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

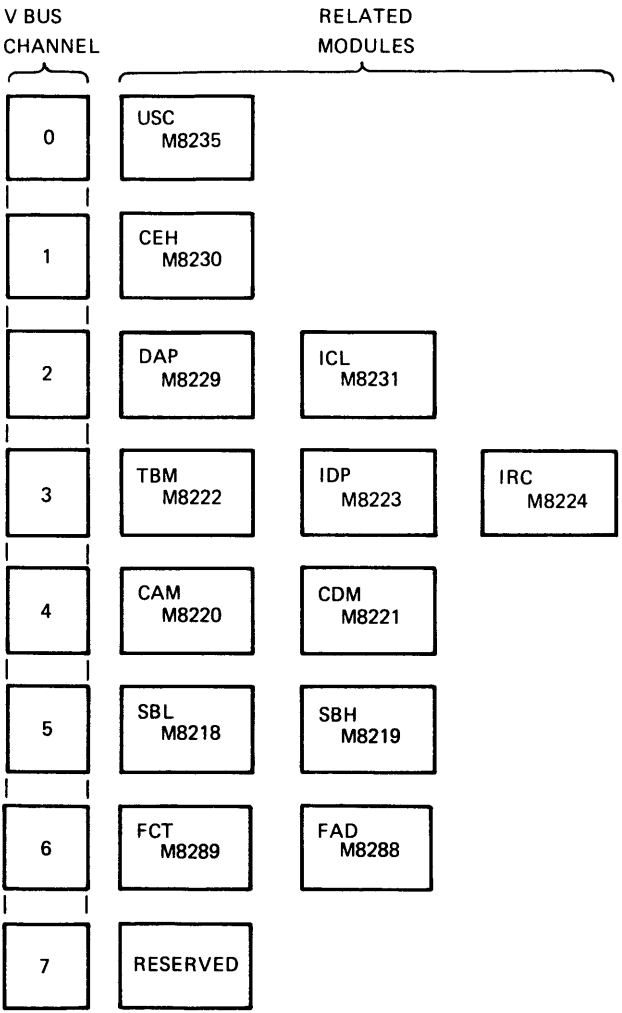
The information on the first error will be in ID[30-39] and U-STACK (trapped microaddresses). Unpredictable on CS parity errors.

The information on the second 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] = ID(2E).

		ID No.
SUMMARY PARA.	TØ	3Ø
CES	T1	31
TRAPPED UPC	T2	32
VA/VIBA	T3	33
D-REG	T4	34
TBERØ	T5	35
TBER1	T6	36
TIME.ADDR	T7	37
PARITY	T8	38
SBI.ERR	T9	39

V-BUS CHANNEL CONFIGURATION



CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
00	0000	000000	USCF	M8235	CPT0	USCF UPCSV 00 H
00	0001	000001	USCF	M8235	CPT0	USCF UPCSV 01 H
00	0002	000002	USCF	M8235	CPT0	USCF UPCSV 02 H
00	0003	000003	USCF	M8235	CPT0	USCF UPCSV 03 H
00	0004	000004	USCF	M8235	CPT0	USCF UPCSV 04 H
00	0005	000005	USCF	M8235	CPT0	USCF UPCSV 05 H
00	0006	000006	USCF	M8235	CPT0	USCF UPCSV 06 H
00	0007	000007	USCF	M8235	CPT0	USCF UPCSV 07 H
00	0008	000010	USCF	M8235	CPT0	USCF UPCSV 08 H
00	0009	000011	USCF	M8235	CPT0	USCF UPCSV 09 H
00	000A	000012	USCF	M8235	CPT0	USCF UPCSV 10 H
00	000B	000013	USCF	M8235	CPT0	USCF UPCSV 11 H
00	000C	000014	USCF	M8235	CPT0	USCF UPCSV 12 H
00	000D	000015	USCB	M8235	CPTX	USCB STALL H
00	000E	000016	USCB	M8235	CPTX	USCB UTRAP H
00	000F	000017	USCJ	M8235	CPTX	USCJ ECO DISPATCH 06 H
00	0010	000020	USCE	M8235	CPT1	USCE ID BUS XCVR EN L
00	0011	000021	USCE	M8235	CPT3	USCE CS WR (31:00) H
00	0012	000022	USCE	M8235	CPT3	USCE CS WR (63:32) H
00	0013	000023	USCE	M8235	CPT3	USCE CS WR (95:64) H
00	0014	000024	USCE	M8235	CPTX	USCE WCS WR CYCLE H
00	0015	000025	USCE	M8235	CPT1	USCE WCS MEM AVAIL L
00	0016	000026	USCL	M8235	CPTX	FCTX ACC OVERRIDE L
00	0017	000027	USCM	M8235	CPTX	USCM IBUF EN (07:00) L
00	0018	000030	USCN	M8235	CPTX	A
00	0019	000031	USCN	M8235	CPTX	ICLK ALU Z (1) H
00	001A	000032	USCN	M8235	CPTX	DCPA LA00 H
00	001B	000033	USCN	M8235	CPTX	D
00	001C	000034	USCN	M8235	CPTX	E
00	001D	000035	USCN	M8235	CPTX	F
00	001E	000036	USCN	M8235	CPTX	ACCA UB0 H
00	001F	000037	USCN	M8235	CPTX	J
00	0020	000040	USCN	M8235	CPTX	K
00	0021	000041	USCN	M8235	CPTX	CEHR PSL C BIT H
00	0022	000042	USCN	M8235	CPTX	ICLK ALU C (1) H
00	0023	000043	USCN	M8235	CPTX	N
00	0024	000044	USCN	M8235	CPTX	P
00	0025	000045	USCN	M8235	CPTX	ACCA UB1 H
00	0026	000046	USCN	M8235	CPTX	S
00	0027	000047	USCN	M8235	CPTX	T
00	0028	000050	USCN	M8235	CPTX	DCPA LA01 H
00	0029	000051	USCN	M8235	CPTX	V
00	002A	000052	USCN	M8235	CPTX	X
00	002B	000053	USCN	M8235	CPTX	Y

V-BUS DIRECTORY (CONT)

00	002C	00054	USCN	M8235	CPTX	ACCA UB2 H
00	002D	00055	USCN	M8235	CPTX	CEHE UTRAP VECT 0 H
00	002E	00056	USCN	M8235	CPTX	TBMD LAST REF CODE 1 H
00	002F	00057	USCN	M8235	CPTX	DAPD SS(1) H
00	0030	00060	USCN	M8235	CPTX	DD
00	0031	00061	USCN	M8235	CPTX	CEHE UTRAP VECT 1 H
00	0032	00062	USCN	M8235	CPTX	TBMD LAST REF CODE 0 H
00	0033	00063	USCN	M8235	CPTX	DDPS SC N.E. 0 H
00	0034	00064	USCN	M8235	CPTX	JJ
00	0035	00065	USCN	M8235	CPTX	CEHE UTRAP VECT 2 H
00	0036	00066	USCN	M8235	CPTX	CEHF NESTED ERR (1) H
00	0037	00067	USCN	M8235	CPTX	ICLK EALU Z (1) H
00	0038	00070	USCN	M8235	CPTX	NN
00	0039	00071	USCN	M8235	CPTX	CEHE UTRAP VECT 3 H
00	003A	00072	USCN	M8235	CPTX	CEHH FPD BIT L
00	003B	00073	USCN	M8235	CPTX	ICLK EALU N (1) H
00	003C	00074	USCN	M8235	CPTX	TT
00	003D	00075	USCN	M8235	CPTX	USCN BEN EN (1B:14) H
00	003E	00076	USCN	M8235	CPTX	USCN BEN EN (1F:1C) H
00	003F	00077	USCN	M8235	CPTX	USCN BEN EN (13:10) H
00	0040	00100	USCN	M8235	CPTX	USCN BEN EN (07:00) H
00	0041	00101	USCN	M8235	CPTX	USCN BEN EN (0F:08) H
00	0042	00102	USCP	M8235	CPTX	USCP BRBIT0(1F:1C) H
00	0043	00103	USCP	M8235	CPTX	ICLE BRBIT0(1B:14) H
00	0044	00104	USCP	M8235	CPTX	ICLE BRBIT0(0F:08) H
00	0045	00105	USCP	M8235	CPTX	USCP BRBIT1(1F:1C) H
00	0046	00106	USCP	M8235	CPTX	ICLE BRBIT1(1B:14) H
00	0047	00107	USCP	M8235	CPTX	ICLE BRBIT1(0F:08) H
00	0048	00110	USCP	M8235	CPTX	USCP BRBIT2(1F:1C) H
00	0049	00111	USCP	M8235	CPTX	ICLE BRBIT2(1B:14) H
00	004A	00112	USCP	M8235	CPTX	ICLE BRBIT2(0F:08) H
00	004B	00113	USCP	M8235	CPTX	USCP BRBIT3(1F:1C) H
00	004C	00114	USCP	M8235	CPTX	ICLE BRBIT3(1B:14) H
00	004D	00115	USCP	M8235	CPTX	USCP BRBIT4(1F:1C) H
00	004E	00116	USCH	M8235	CPT3	USCH SYNC PULSE H
00	004F	00117	USCJ	M8235	CPT0	CIBN D MAINT RTN H
00	0050	00120	USCJ	M8235	CPTX	USCJ INIT (1) H
00	0051	00121	USCJ	M8235	CPTX	USCJ STALL (1) H
00	0052	00122	USCJ	M8235	CPTX	USCJ UTRAP (1) H
00	0053	00123	USCJ	M8235	CPTX	USCJ UECO (1) H
00	0054	00124	USCJ	M8235	CPTX	USCJ MAINT RET (1) H
00	0055	00125	USCJ	M8235	CPTX	USCJ PRIOR 0 L
00	0056	00126	USCJ	M8235	CPTX	USCJ PRIOR 1 L
00	0057	00127	USCJ	M8235	CPTX	USCJ PRIOR 2 L

V-BUS DIRECTORY (CONT)

00	0058	00130	USCM	M8235	CPT2	USCM BUF UPC 00 H
00	0059	00131	USCM	M8235	CPT2	USCM BUF UPC 01 H
00	005A	00132	USCM	M8235	CPT2	USCM BUF UPC 02 H
00	005B	00133	USCM	M8235	CPT2	USCM BUF UPC 03 H
00	005C	00134	USCM	M8235	CPT2	USCM BUF UPC 04 H
00	005D	00135	USCM	M8235	CPT2	USCM BUF UPC 05 H
00	005E	00136	USCM	M8235	CPT2	USCM BUF UPC 06 H
00	005F	00137	USCM	M8235	CPT2	USCM BUF UPC 07 H
00	0060	00140	USCM	M8235	CPT2	USCM BUF UPC 08 H
00	0061	00141	USCM	M8235	CPT2	USCM BUF UPC 09 H
00	0062	00142	USCM	M8235	CPT2	USCM BUF UPC 10 H
00	0063	00143	USCM	M8235	CPT2	USCM BUF UPC 11 H
00	0064	00144	USCM	M8235	CPT2	USCM BUF UPC 12 H
00	0065	00145	USCX	M8235	CPTX	RESERVED
00	0066	00146	USCX	M8235	CPTX	RESERVED
00	0067	00147	USCX	M8235	CPTX	RESERVED

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
01	0000	00000	CEME	M8230	CPTX	PCSC PAR ERR (95:64) H
01	0001	00001	CEME	M8230	CPTX	PCSC PAR ERR (63:32) H
01	0002	00002	CEME	M8230	CPTX	PCSC PAR ERR (31:00) H
01	0003	00003	CEME	M8230	CPTX	SRLP PAR ERR TRAP L
01	0004	00004	CEME	M8230	CPTX	TRMW PROT UTRAP L
01	0005	00005	CEMF	M8230	CPTX	CEMF IRD STATE H
01	0006	00006	CEMF	M8230	CPTX	CEMF READ RLOG H
01	0007	00007	CEMF	M8230	CPTX	CEMF LOAD STATE H
01	0008	00010	CEMF	M8230	CPTX	CEMF CLR UWORD (1) H
01	0009	00011	CEMP	M8230	CPTX	ICLS EN ID XCEIV L
01	000A	00012	CEMA	M8230	CPTX	DEPM BMX31 L
01	000B	00013	CEMA	M8230	CPTX	DDPD BMX15 L
01	000C	00014	CEMA	M8230	CPTX	DCPD BMX07 L
01	000D	00015	CEMA	M8230	CPTX	DEPD AMX31 L
01	000E	00016	CEMA	M8230	CPTX	DDPB AMX15 L
01	000F	00017	CEMA	M8230	CPTX	DCPD AMX07 L
01	0010	00020	CEMA	M8230	CPTX	DEPK ALU BUFF31 L
01	0011	00021	CEMA	M8230	CPTX	DCPL ALU CARRY31 L
01	0012	00022	CEMA	M8230	CPTX	DCPL ALU CARRY15 L
01	0013	00023	CEMA	M8230	CPTX	DCPL ALU CARRY07 L
01	0014	00024	CEMA	M8230	CPTX	CEMA ALU BUFF17 L
01	0015	00025	CEMA	M8230	CPTX	CEMA ALU BUFF16 L
01	0016	00026	CEMA	M8230	CPTX	CEMA ALU BUFF15 L
01	0017	00027	CEMA	M8230	CPTX	CEMA ALU BUFF07 L
01	0018	00030	CEMA	M8230	CPTX	DEPN ALU(30:18)=0 L
01	0019	00031	CEMA	M8230	CPTX	DDPF ALU(15:08)=0 L
01	001A	00032	CEMA	M8230	CPTX	DCPF ALU(07:00)=0 L
01	001B	00033	CEMA	M8230	CPTX	BUS ALU BYTE2,3 A=B H
01	001C	00034	CEMA	M8230	CPTX	BUS ALU BYTE1 A=B H
01	001D	00035	CEMA	M8230	CPTX	BUS ALU BYTE0 A=B H
01	001E	00036	CEMB	M8230	CPTX	DCPA AMX00 L
01	001F	00037	CEMB	M8230	CPTX	DAPB AUALU=A PLUS B L
01	0020	00040	CEMB	M8230	CPTX	DAPB AUALU=A MINUS B L
01	0021	00041	CEMC	M8230	CPTX	DDPN EALU09 H
01	0022	00042	CEMC	M8230	CPTX	DDPN EALU08 H
01	0023	00043	CEMC	M8230	CPTX	ACCX NDATA H
01	0024	00044	CEMC	M8230	CPTX	ACCX ZDATA H
01	0025	00045	CEMC	M8230	CPTX	ACCX VDATA H
01	0026	00046	CEMC	M8230	CPTX	ACCX CDATA H
01	0027	00047	CEMD	M8230	CPTX	CEMD SECOND REF H
01	0028	00050	CEMD	M8230	CPTX	SBLT STALL L
01	0029	00051	CEMD	M8230	CPTX	IRCJ DQ CONT H
01	002A	00052	CEMD	M8230	CPTX	IRCJ FLOAT H
01	002B	00053	CEMD	M8230	CPTX	IRCJ WORD CONT H

V-BUS DIRECTORY (CONT)

01	002C	00054	CEMD	M8230	CPTX	IRCJ BYTE CONT H
01	002D	00055	CEMD	M8230	CPTX	TBMW SAVE CONTEXT H
01	002E	00056	CEME	M8230	CPTX	DDPS FLOAT NZERO H
01	002F	00057	CEME	M8230	CPTX	USCB CLR UTRAP L
01	0030	00060	CEMR	M8230	CPTX	DCPH VA02(1) H
01	0031	00061	CEMR	M8230	CPTX	DCPJ VA01(1) H
01	0032	00062	CEMR	M8230	CPTX	DCPJ VA00(1) H
01	0033	00063	CEME	M8230	CPTX	TBMW EN CMODADRS H
01	0034	00064	CEME	M8230	CPTX	TBMN PAGE EDGE H
01	0035	00065	CEME	M8230	CPTX	TBMW EN UNALIGN TRAP H
01	0036	00066	CEME	M8230	CPTX	SBLM TIMEOUT TRAP L
01	0037	00067	CEME	M8230	CPTX	SBLR RDS TRAP L
01	0038	00070	CEME	M8230	CPTX	TBMW TB PAR UTRAP L
01	0039	00071	CEME	M8230	CPTX	TBMW MISS UTRAP L
01	003A	00072	CEME	M8230	CPTX	TBMW MBIT UTRAP L
01	003B	00073	CEME	M8230	CPTX	CEME CS PE TRAP H
01	003C	00074	CEMX	M8230	CPTX	RESERVED
01	003D	00075	CEMX	M8230	CPTX	RESERVED
01	003E	00076	CEMX	M8230	CPTX	RESERVED
01	003F	00077	CEMX	M8230	CPTX	RESERVED

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
02	0000	00000	DAPA	M8229	CPTX	IRCF OPC0 H
02	0001	00001	DAPA	M8229	CPTX	IRCF OPC1 H
02	0002	00002	DAPA	M8229	CPTX	IRCF OPC2 H
02	0003	00003	DAPA	M8229	CPTX	IRCF OPC3 H
02	0004	00004	DAPA	M8229	CPTX	IRCF OPC4 H
02	0005	00005	DAPA	M8229	CPTX	IRCF OPC5 H
02	0006	00006	DAPA	M8229	CPTX	IRCF OPC6 H
02	0007	00007	DAPA	M8229	CPTX	IRCF OPC7 H
02	0008	00010	DAPX	M8229	CPTX	CEHD MEMREF DT=B H
02	0009	00011	DAPX	M8229	CPTX	CEHD MEMREF DT=LFDG H
02	000A	00012	DAPX	M8229	CPTX	RESERVED
02	0008	00013	DAPC	M8229	CPTX	RESERVED
02	000C	00014	DAPC	M8229	CPTX	IRCE BYTE CONT H
02	000D	00015	DAPC	M8229	CPTX	IRCE WORD CONT H
02	000E	00016	DAPC	M8229	CPTX	IRCE LFDQ CONT H
02	000F	00017	DAPD	M8229	CPTX	IRCH PC REG H
02	0010	00020	DAPF	M8229	CPTX	RESERVED
02	0011	00021	DAPF	M8229	CPTX	IRCE SP1 CON2 H
02	0012	00022	DAPF	M8229	CPTX	IRCE SP1 CON1 H
02	0013	00023	DAPF	M8229	CPTX	IRCE SP1 CON0 H
02	0014	00024	DAPX	M8229	CPTX	DAPB RLOG UPDATE H
02	0015	00025	DAPD	M8229	CPTX	CEHF READ RLOG H
02	0016	00026	DAPF	M8229	CPTX	RESERVED
02	0017	00027	DPAF	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	IDPN SP1 ADR0 L
02	001C	00034	DAPL	M8229	CPTX	IDPN SP1 ADR1 L
02	001D	00035	DAPL	M8229	CPTX	IDPN SP1 ADR2 L
02	001E	00036	DAPL	M8229	CPTX	IDPN SP1 ADR3 L
02	001F	00037	DAPL	M8229	CPTX	IDPN SP2 ADR0 L
02	0020	00040	DAPL	M8229	CPTX	IDPN SP2 ADR1 L
02	0021	00041	DAPL	M8229	CPTX	IDPN SP2 ADR2 L
02	0022	00042	DAPL	M8229	CPTX	IDPN SP2 ADR3 L
02	0023	00043	DAPL	M8229	CPTX	IDPN PRN 0 L
02	0024	00044	DAPL	M8229	CPTX	IDPN PRN 1 L
02	0025	00045	DAPL	M8229	CPTX	IDPN PRN 2 L
02	0026	00046	DAPL	M8229	CPTX	IDPN PRN 3 L
02	0027	00047	DAPX	M8229	CPTX	RESERVED

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
02	0028	00050	ICLT	M8231	CPTX	WCSC WCS EVEN PAR H
02	0029	00051	ICLA	M8231	CPTX	SBLM TIMO CNF INTR L
02	002A	00052	ICLA	M8231	CPTX	SBHL FAULT INTR H
02	002B	00053	ICLA	M8231	CPTX	SBHE SBI ALERT R H
02	002C	00054	ICLA	M8231	CPTX	SBLM CRD RDS INTR L
02	002D	00055	ICLA	M8231	CPTX	SBHK COMP INTR H
02	002E	00056	ICLA	M8231	CPTX	SBHE SBI REQ7 R H
02	002F	00057	ICLA	M8231	CPTX	SBHE SBI REQ6 R H
02	0030	00060	ICLA	M8231	CPTX	SBHE SBI REQ5 R H
02	0031	00061	ICLA	M8231	CPTX	SBHE SBI REQ4 R H
02	0032	00062	ICLB	M8231	CPTX	ICLB IPL ACT 4 H
02	0033	00063	ICLB	M8231	CPTX	ICLB IPL ACT 3 H
02	0034	00064	ICLB	M8231	CPTX	ICLB IPL ACT 2 H
02	0035	00065	ICLB	M8231	CPTX	ICLB IPL ACT 1 H
02	0036	00066	ICLB	M8231	CPTX	ICLB IPL ACT 0 H
02	0037	00067	ICLC	M8231	CPTX	CEHJ PRIOR 3 H
02	0038	00070	ICLC	M8231	CPTX	CEHJ PRIOR 2 H
02	0039	00071	ICLC	M8231	CPTX	CEHJ PRIOR 1 H
02	003A	00072	ICLC	M8231	CPTX	CEHJ PRIOR 0 H
02	003B	00073	ICLC	M8231	CPTX	CEHR INTR REQ L
02	003C	00074	ICLC	M8231	CPTX	CIBS CNSL RCV INTR H
02	003D	00075	ICLC	M8231	CPTX	CIBS CNSL XMIT INTR H
02	003E	00076	ICLD	M8231	CPTX	CEHC TRAP CODE2 (1) H
02	003F	00077	ICLD	M8231	CPTX	CEHC TRAP CODE1 (1) H
02	0040	00100	ICLD	M8231	CPTX	CEHC TRAP CODE0 (1) H
02	0041	00101	ICLD	M8231	CPTX	CEHP ID30 H
02	0042	00102	ICLD	M8231	CPTX	IRCE STALL+SVC L
02	0043	00103	ICLD	M8231	CPTX	CIBN HALT REQ H
02	0044	00104	ICLD	M8231	CPTX	RESERVED
02	0045	00105	ICLE	M8231	CPTX	DDPS BRANCH3 H
02	0046	00106	ICLE	M8231	CPTX	DBPV BR BIT3 H
02	0047	00107	ICLE	M8231	CPTX	DDPS BRANCH2 H
02	0048	00110	ICLE	M8231	CPTX	DBPV BR BIT2 H
02	0049	00111	ICLE	M8231	CPTX	DDPS BRANCH1 H
02	004A	00112	ICLE	M8231	CPTX	DBPV BR BIT1 H
02	004B	00113	ICLE	M8231	CPTX	DDPS BRANCH0 H
02	004C	00114	ICLE	M8231	CPTX	DBPV BR BIT0 H
02	004D	00115	ICLE	M8231	CPTX	IRCH BRC1 H
02	004E	00116	ICLE	M8231	CPTX	IRCH BRC0 H
02	004F	00117	ICLE	M8231	CPTX	IRCF OPC 0 H
02	0050	00120	ICL?	M8231	CPTX	IRCH READ OP H
02	0051	00121	ICL?	M8231	CPTX	RESERVED
02	0052	00122	ICLE	M8231	CPTX	ICLE REM BEMX S2 H
02	0053	00123	ICLE	M8231	CPTX	ICLE REM BEMX S1 H

V-BUS DIRECTORY (CONT)

02	0054	00124	ICLE	M8231	CPTX	ICLE REM BEMX $0 H
02	0055	00125	ICLH	M8231	CPTX	ICLH ID TO PSL H
02	0056	00126	ICLH	M8231	CPTX	ICLH ID TO VECT L
02	0057	00127	ICLH	M8231	CPTX	ICLK ID TO CES H
02	0058	00130	ICLH	M8231	CPTY	ICLH ID TO ATMP L
02	0059	00131	ICLH	M8231	CPTX	ICLH ID TO BTMP L
02	005A	00132	ICLH	M8231	CPTX	ICLH IDM $2 L
02	005B	00133	ICLH	M8231	CPTX	ICLH IDM $1 L
02	005C	00134	ICLH	M8231	CPTX	ICLH IDM $0 L
02	005D	00135	ICLJ	M8231	CPTX	DDPR SC05 (1) H
02	005E	00136	ICLJ	M8231	CPTX	DDPR SC04 (1) H
02	005F	00137	ICLJ	M8231	CPTX	DDPR SC03 (1) H
02	0060	00140	ICLJ	M8231	CPTX	DDPR SC02 (1) H
02	0061	00141	ICLJ	M8231	CPTX	DDPR SC01 (1) H
02	0062	00142	ICLJ	M8231	CPTX	DDPR SC00 (1) H
02	0063	00143	ICLJ	M8231	CPTX	ICLJ D TO ID L
02	0064	00144	ICLJ	M8231	CPTX	ICLJ ID TO Q L
02	0065	00145	ICLK	M8231	CPTX	DDPN EALU09 H
02	0066	00146	ICLK	M8231	CPTX	DDPN EALU00 L
02	0067	00147	ICLX	M8231	CPTX	RESERVED

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
03	0000	00000	TBMD	M8222	CPTX	TBMD D TO MD L
03	0001	00001	TBMD	M8222	CPTX	TBMD MASK TO MD L
03	0002	00002	TBMD	M8222	CPTX	TBMD EN ID DRIVERS L
03	0003	00003	TBMS	M8222	CPTX	TBMS CPT0 L
03	0004	00004	TBMF	M8222	CPTX	TBMF GRP 0 WP L
03	0005	00005	TBMF	M8222	CPTX	TBMF GRP 1 WP L
03	0006	00006	TBMC	M8222	CPTX	CAMU TB GRP 0 MATCH H
03	0007	00007	TBMC	M8222	CPTX	CAMU TB GRP 1 MATCH H
03	0008	00010	TBMU	M8222	CPTX	CEHE CMODD ADRS TRAP L
03	0009	00011	TBMU	M8222	CPTX	CEHE PAGE TRAP H
03	000A	00012	TBMW	M8222	CPTX	CEHE CS PAR ERR H
03	000B	00013	TBMX	M8222	CPTX	RESERVED
03	000C	00014	TBMN	M8222	CPTX	USCR ABORT CYCLE H
03	000D	00015	TBMN	M8222	CPTX	IRCH IB WRITE CHK H
03	000E	00016	TBMX	M8222	CPTX	RESERVED
03	000F	00017	TBMU	M8222	CPTX	TBMU CANCEL L
03	0010	00020	TBMK	M8222	CPTX	SBLB SBI PA 09 L
03	0011	00021	TBMK	M8222	CPTX	SBLB SBI PA 10 L
03	0012	00022	TBMK	M8222	CPTX	SBLB SBI PA 11 L
03	0013	00023	TBMC	M8222	CPTX	TBMC ENABLE IA H
03	0014	00024	TBMX	M8222	CPTX	RESERVED
03	0015	00025	TBMX	M8222	CPTX	RESERVED
03	0016	00026	TBMX	M8222	CPTX	SBLS IB ERR LTH H
03	0017	00027	TBMW	M8222	CPTX	SBLT STALL L
03	0018	00030	TBMX	M8222	CPTX	RESERVED
03	0019	00031	TBMX	M8222	CPTX	RESERVED
03	001A	00032	TBMX	M8222	CPTX	RESERVED
03	001B	00033	TBMD	M8222	CPTX	CAMV MODIFY L
03	001C	00034	TBMB	M8222	CPTX	CAMV PROTECT CODE 0 L
03	001D	00035	TBMB	M8222	CPTX	CAMV PROTECT CODE 1 L
03	001E	00036	TBMB	M8222	CPTX	CAMV PROTECT CODE 2 L
03	001F	00037	TBMB	M8222	CPTX	CAMV PROTECT CODE 3 L
03	0020	00040	IDPA	M8223	CPT0	IDPA BUF B0-7(1) H
03	0021	00041	IDPA	M8223	CPT0	IDPA BUF B0-6(1) H
03	0022	00042	IDPA	M8223	CPT0	IDPA BUF B0-5(1) H
03	0023	00043	IDPA	M8223	CPT0	IDPA BUF B0-4(1) H
03	0024	00044	IDPA	M8223	CPT0	IDPA BUF B0-3(1) H
03	0025	00045	IDPA	M8223	CPT0	IDPA BUF B0-2(1) H
03	0026	00046	IDPA	M8223	CPT0	IDPA BUF B0-1(1) H
03	0027	00047	IDPA	M8223	CPT0	IDPA BUF B0-0(1) H

V-BUS DIRECTORY (CONT)

03	0028	00050	IDPA	M8223	CPT0	IDPA BUF B1=7(1) H
03	0029	00051	IDPA	M8223	CPT0	IDPA BUF B1=6(1) H
03	002A	00052	IDPA	M8223	CPT0	IDPA BUF B1=5(1) H
03	002B	00053	IDPA	M8223	CPT0	IDPA BUF B1=4(1) H
03	002C	00054	IDPA	M8223	CPT0	IDPA BUF B1=3(1) H
03	002D	00055	IDPA	M8223	CPT0	IDPA BUF B1=2(1) H
03	002E	00056	IDPA	M8223	CPT0	IDPA BUF B1=1(1) H
03	002F	00057	IDPA	M8223	CPT0	IDPA BUF B1=0(1) H
03	0030	00060	IDPH	M8223	CPT0	IDPH IBC 3(1) H
03	0031	00061	IDPH	M8223	CPT0	IDPH IBC 2(1) H
03	0032	00062	IDPH	M8223	CPT0	IDPH IBC 1(1) H
03	0033	00063	IDPH	M8223	CPT0	IDPH IBC 0(1) H
03	0034	00064	IDPH	M8223	CPTX	IDPH CLR 0 L
03	0035	00065	IDPH	M8223	CPTX	IRCE SAVE H
03	0036	00066	IDPA	M8223	CPT0	IDPA VAX H
03	0037	00067	IDPA	M8223	CPTX	IDPA DST R MODE H
03	0038	00070	IDPJ	M8223	CPTX	SBLR IB READ DATA L
03	0039	00071	IDPX	M8223	CPTX	RESERVED
03	003A	00072	IDPJ	M8223	CPT0	IDPJ COUNT H
03	003B	00073	IDPJ	M8223	CPTX	IDPJ FLUSH L
03	003C	00074	IDPJ	M8223	CPTX	IDPJ B5 VAL(1) H
03	003D	00075	IDPJ	M8223	CPTX	IDPJ B4 VAL(1) H
03	003E	00076	IDPJ	M8223	CPTX	IDPJ B3 VAL(0) H
03	003F	00077	IDPJ	M8223	CPTX	IDPJ B2 VAL(0) H
03	0040	00100	IDPJ	M8223	CPTX	IRCD PC MODE H
03	0041	00101	IDPM	M8223	CPTX	IRCD SEL LONG L
03	0042	00102	IDPM	M8223	CPTX	IRCD SEL WORD L
03	0043	00103	IDPM	M8223	CPTX	IRCD SEL BYTE H
03	0044	00104	IDPM	M8223	CPTX	IRCE CTX 3 L
03	0045	00105	IDPM	M8223	CPTX	IRCE CTX 2 L
03	0046	00106	IDPL	M8223	CPTX	IDPL ID BUS XCVR EN L
03	0047	00107	IDPX	M8223	CPTX	RESERVED
03	0048	00110	IDPM	M8223	CPTX	IDPM 8 DELTA PC 2 H
03	0049	00111	IDPM	M8223	CPTX	IDPM 16 BIT 8 DEST L
03	004A	00112	IDPM	M8223	CPTX	IDPM 8 DEST H
03	004B	00113	IDPM	M8223	CPTX	IDPM B DELTA PC 1 H
03	004C	00114	IDPM	M8223	CPTX	IDPM VAXSL L
03	004D	00115	IDPM	M8223	CPTX	IDPM B DELTA PC 0 H
03	004E	00116	IDPM	M8223	CPTX	TBMX VA 01 H
03	004F	00117	IDPM	M8223	CPTX	TBMX VA 00 H
03	0050	00120	IRCH	M8224	CPTX	TBMX TB ERR L
03	0051	00121	IRCH	M8224	CPTX	TBMX TB MISS L
03	0052	00122	IRCC	M8224	CPTX	CEHH FPD BIT L
03	0053	00123	IRCX	M8224	CPTX	RESERVED

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
03	0054	00124	IRCE	M8224	CPTX	IRCE IB ADVANCE H
03	0055	00125	IRCJ	M8224	CPTX	IRCJ SP2 CON 1 H
03	0056	00126	IRCJ	M8224	CPTX	IRCJ SP2 CON 0 H
03	0057	00127	IRCM	M8224	CPTX	IRCM DATA EN L
03	0058	00130	IRCE	M8224	CPTO	ICLD SERVICE H
03	0059	00131	IRCE	M8224	CPTO	ICLD SERVICE BIT 0 H
03	005A	00132	IRCE	M8224	CPTO	ICLD SERVICE BIT 1 H
03	005B	00133	IRCE	M8224	CPTO	ICLD SERVICE BIT 2 H
03	005C	00134	IRCC	M8224	CPTO	IRCC EXEC CT 0 H
03	005D	00135	IRCC	M8224	CPTO	IRCC EXEC CT 1 H
03	005E	00136	IRCC	M8224	CPTO	IRCC EXEC CT 2 H
03	005F	00137	IRCX	M8224	CPTO	RESERVED

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
04	0000	00000	CAMP	M8220	CPTX	TBMX FORCE ERR 2 L
04	0001	00001	CAMP	M8220	CPTX	TBMX FORCE ERR 1 L
04	0002	00002	CAMP	M8220	CPTX	TBMX FORCE ERR 0 L
04	0003	00003	CAMB	M8220	CPTX	SBHF REV PAR FIELD 3 H
04	0004	00004	CAMB	M8220	CPTX	SBHF REV PAR FIELD 2 H
04	0005	00005	CAMB	M8220	CPTX	SBHF REV PAR FIELD 1 H
04	0006	00007	CAMB	M8220	CPTX	SBHF REV PAR FIELD 0 H
04	0007	00007	CAMS	M8220	CPTX	CAMS G0 ADR PAR 2 OD H
04	0008	00010	CAMS	M8220	CPTX	CAMS G0 ADR PAR 1 OD H
04	0009	00011	CAMS	M8220	CPTX	CAMS G0 ADR PAR 0 OD H
04	000A	00012	CAMT	M8220	CPTX	CAMT G1 ADR PAR 2 OD H
04	0008	00013	CAMT	M8220	CPTX	CAMT G1 ADR PAR 1 OD H
04	000C	00014	CAMT	M8220	CPTX	CAMT G1 ADR PAR 0 OD H
04	000D	00015	CAMV	M8220	CPTX	CAMV TB PAR 2 H
04	000E	00016	CAMU	M8220	CPTX	CAMU TB PAR 1 H
04	000F	00017	CAMU	M8220	CPTX	CAMU TB PAR 0 H
04	0010	00020	CAMK	M8220	CPTX	CAMK G1 MATCH H
04	0011	00021	CAMK	M8220	CPTX	CAMK G0 MATCH H
04	0012	00022	CAMP	M8220	CPTX	SBLN SBI MISS DATA G1 H
04	0013	00023	CAMP	M8220	CPTX	SBLN SBI MISS DATA G0 H
04	0014	00024	CAMM	M8220	CPT3	CAMM CPT3 B H
04	0015	00025	CAMM	M8220	CPT2	CAMM CPT2 B H
04	0016	00026	CAMM	M8220	CPT1	CAMM CPT1 B L
04	0017	00027	CAMM	M8220	CPT1	CAMM CPT1 B H
04	0018	00030	CAMP	M8220	CPTX	CAMP G1 WRITE ENABLE H
04	0019	00031	CAMP	M8220	CPTX	CAMP G0 WRITE ENABLE H
04	001A	00032	CAMP	M8220	CPTX	SBHN FORCE MISS G1 H
04	0018	00033	CAMP	M8220	CPTX	SBHF FORCE MISS G0 H
04	001C	00034	CAMX	M8220	CPTX	RESERVED
04	001D	00035	CAMB	M8220	CPTX	CAMB LATCH VALID BIT H
04	001E	00036	CAMX	M8220	CPTX	TBMX FORCE ERR 3 L
04	001F	00037	CAMB	M8220	CPTX	SBHF REV PAR FIELD 3 L
04	0020	00040	CAML	M8220	CPTX	CAML G1 BYTE 2 PAR OD H
04	0021	00041	CAML	M8220	CPTX	CAML G1 BYTE 2 PAR EV H
04	0022	00042	CAML	M8220	CPTX	CAML G1 BYTE 1 PAR OD H
04	0023	00043	CAML	M8220	CPTX	CAML G1 BYTE 1 PAR EV H
04	0024	00044	CAML	M8220	CPTX	CAML G1 BYTE 0 PAR OD H
04	0025	00045	CAML	M8220	CPTX	CAML G1 BYTE 0 PAR EV H
04	0026	00046	CAML	M8220	CPTX	CAML G0 BYTE 2 PAR OD H
04	0027	00047	CAML	M8220	CPTX	CAML G0 BYTE 2 PAR EV H

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
04	0028	00050	CAML	M8220	CPTX	CAML G0 BYTE 1 PAR OD H
04	0029	00051	CAML	M8220	CPTX	CAML G0 BYTE 1 PAR EV H
04	002A	00052	CAML	M8220	CPTX	CAML G0 BYTE 0 PAR OD H
04	002B	00053	CAML	M8220	CPTX	CAML G0 BYTE 0 PAR EV H
04	002C	00054	CAMB	M8220	CPTX	CAMB TAG PAR 2 EVEN H
04	002D	00055	CAMB	M8220	CPTX	CAMB TAG PAR 1 EVEN H
04	002E	00056	CAMB	M8220	CPTX	CAMB TAG PAR 0 EVEN H
04	002F	00057	CAMB	M8220	CPT1	CAMB PA LATCH 12 H
04	0030	00060	CAMB	M8220	CPT1	CAMB PA LATCH 13 H
04	0031	00061	CAMB	M8220	CPT1	CAMB PA LATCH 14 H
04	0032	00062	CAMB	M8220	CPT1	CAMB PA LATCH 15 H
04	0033	00063	CAMB	M8220	CPT1	CAMB PA LATCH 16 H
04	0034	00064	CAMB	M8220	CPT1	CAMB PA LATCH 17 H
04	0035	00065	CAMB	M8220	CPT1	CAMB PA LATCH 18 H
04	0036	00066	CAMB	M8220	CPT1	CAMB PA LATCH 19 H
04	0037	00067	CAMB	M8220	CPT1	CAMB PA LATCH 20 H
04	0038	00070	CAMB	M8220	CPT1	CAMB PA LATCH 21 H
04	0039	00071	CAMB	M8220	CPT1	CAMB PA LATCH 22 H
04	003A	00072	CAMB	M8220	CPT1	CAMB PA LATCH 23 H
04	003B	00073	CAMB	M8220	CPT1	CAMB PA LATCH 24 H
04	003C	00074	CAMB	M8220	CPT1	CAMB PA LATCH 25 H
04	003D	00075	CAMB	M8220	CPT1	CAMB PA LATCH 26 H
04	003E	00076	CAMB	M8220	CPT1	CAMB PA LATCH 27 H
04	003F	00077	CAMB	M8220	CPT1	CAMB PA LATCH 28 H
04	0040	00100	CDMX	M8221	CPTX	RESERVED
04	0041	00101	CDMX	M8221	CPTX	RESERVED
04	0042	00102	CDMX	M8221	CPTX	RESERVED
04	0043	00103	CDMU	M8221	CPT2	CDMU CPT2 H
04	0044	00104	CDMU	M8221	CPT1	RESERVED
04	0045	00105	CDMU	M8221	CPT1	RESERVED
04	0046	00106	CDMU	M8221	CPT1	CDMU CPT1 A L
04	0047	00107	CDMU	M8221	CPT1	CDMU CPT1 A H
04	0048	00110	CDMT	M8221	CPTX	TBMD EN CDM DATA L
04	0049	00111	CDMS	M8221	CPTX	CDMS G1 B3 PAR ODD H
04	004A	00112	CDMS	M8221	CPTX	CDMS G1 B3 PAR EVEN H
04	004B	00113	CDMS	M8221	CPTX	CDMS G1 B2 PAR ODD H
04	004C	00114	CDMS	M8221	CPTX	CDMS G1 B2 PAR EVEN H
04	004D	00115	CDMS	M8221	CPTX	CDMS G1 B1 PAR ODD H
04	004E	00116	CDMS	M8221	CPTX	CDMS G1 B1 PAR EVEN H
04	004F	00117	CDMS	M8221	CPTX	CDMS G1 B0 PAR ODD H

V-BUS DIRECTORY (CONT)

04	0050	00120	CDMS	M8221	CPTX	CDMS G1 B0 PAR EVEN H
04	0051	00121	CDMR	M8221	CPTX	CDMR G0 B3 PAR ODD H
04	0052	00122	CDMR	M8221	CPTX	CDMR G0 B3 PAR EVEN H
04	0053	00123	CDMR	M8221	CPTX	CDMR G0 B2 PAR ODD H
04	0054	00124	CDMR	M8221	CPTX	CDMR G0 B2 PAR EVEN H
04	0055	00125	CDMR	M8221	CPTX	CDMR G0 B1 PAR ODD H
04	0056	00126	CDMR	M8221	CPTX	CDMR G0 B1 PAR EVEN H
04	0057	00127	CDMR	M8221	CPTX	CDMR G0 B0 PAR ODD H
04	0058	00130	CDMR	M8221	CPTX	CDMR G0 B0 PAR EVEN H
04	0059	00131	CDMX	M8221	CPTX	RESERVED
04	005A	00132	CDMX	M8221	CPTX	RESERVED
04	0058	00133	CDMH	M8221	CPT1	CDMH ADDR LATCH 11 H
04	005C	00134	CDMH	M8221	CPT1	CDMH ADDR LATCH 10 H
04	005D	00135	CDMH	M8221	CPT1	CDMH ADDR LATCH 9 H
04	005E	00136	CDMH	M8221	CPT1	CDMH ADDR LATCH 8 H
04	005F	00137	CDMH	M8221	CPT1	CDMH ADDR LATCH 7 H
04	0060	00140	CDMH	M8221	CPT1	CDMH ADDR LATCH 6 H
04	0061	00141	CDMH	M8221	CPT1	CDMH ADDR LATCH 5 H
04	0062	00142	CDMH	M8221	CPT1	CDMH ADDR LATCH 4 H
04	0063	00143	CDMH	M8221	CPT1	CDMH ADDR LATCH 3 H
04	0064	00144	CDMH	M8221	CPT1	CDMH ADDR LATCH 2 H
04	0065	00145	CDMB	M8221	CPTX	SBHF REV PAR 3 L
04	0066	00146	CDMB	M8221	CPTX	SBHF REV PAR 2 L
04	0067	00147	CDMB	M8221	CPTX	SBHF REV PAR 1 L
04	0068	00150	CDMB	M8221	CPTX	SBHF REV PAR 0 L
04	0069	00151	CDMB	M8221	CPT3	CAMP G1 WRITE ENABLE H
04	006A	00152	CDMB	M8221	CPT3	CAMP G0 WRITE ENABLE H
04	006B	00153	CDMX	M8221	CPTX	RESERVED
04	006C	00154	CDMA	M8221	CPT2	CDMA MASK 3 H
04	006D	00155	CDMA	M8221	CPT2	CDMA MASK 2 H
04	006E	00156	CDMA	M8221	CPT2	CDMA MASK 1 H
04	006F	00157	CDMA	M8221	CPT2	CDMA MASK 0 H

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
05	0000	00000	SBLH	M8218	CPTX	SBHP EN ID DRIVERS L
05	0001	00001	SBLF	M8218	CPTX	SBHP ID ADDR 2 L
05	0002	00002	SBLF	M8218	CPTX	SBHP ID ADDR 1 L
05	0003	00003	SBLE	M8218	CPTX	TBMC ENABLE IA H
05	0004	00004	SBLS	M8218	CPTX	SBLS ADRS LATCH 29 H
05	0005	00005	SBLS	M8218	CPTX	IDPJ IB REQ H
05	0006	00006	SBLP	M8218	CPTX	SBLP MD TO D L
05	0007	00007	SBLF	M8218	CPTX	SBHP ID ADDR 0 L
05	0008	00010	SBLM	M8218	CPTX	SBHN CRD L
05	0009	00011	SBLP	M8218	CPTX	TBMN BUF UMCT 0 L
05	000A	00012	SBLP	M8218	CPTX	TBMN BUF UMCT 1 L
05	000B	00013	SBLP	M8218	CPTX	TBMN BUF UMCT 2 L
05	000C	00014	SBLP	M8218	CPTX	TBMN BUF UMCT 3 L
05	000D	00015	SBLP	M8218	CPTX	TBMN BUF UADS L
05	000E	00016	SBLP	M8218	CPTX	TBMN BUF UFS L
05	000F	00017	SBLM	M8218	CPTX	SBHN RDS L
05	0010	00020	SBLR	M8218	CPTX	SBHM SET INVALID L
05	0011	00021	SBLE	M8218	CPTX	SBHM SET SBI CYCLE H
05	0012	00022	SBLL	M8218	CPTX	SBHR SEND DATA H
05	0013	00023	SBLE	M8218	CPTX	SBHM ANY READ DATA L
05	0014	00024	SBLK	M8218	CPTX	SBLK LATCH TIMO REG L
05	0015	00025	SBLD	M8218	CPTX	TBMU CANCEL L
05	0016	00026	SBLW	M8218	CPTX	CLKL SYS INIT B L
05	0017	00027	SBLR	M8218	CPTX	SBLR FORCE SBI L
05	0018	00030	SBLX	M8218	CPTX	RESERVED
05	0019	00031	SBLX	M8218	CPTX	RESERVED
05	001A	00032	SBLC	M8218	CPTX	SBLC WRITE DATA 00 H
05	001B	00033	SBLC	M8218	CPTX	SBLC WRITE DATA 01 H
05	001C	00034	SBLC	M8218	CPTX	SBLC WRITE DATA 02 H
05	001D	00035	SBLC	M8218	CPTX	SBLC WRITE DATA 03 H
05	001E	00036	SBLC	M8218	CPTX	SBLC WRITE DATA 04 H
05	001F	00037	SBLC	M8218	CPTX	SBLC WRITE DATA 05 H
05	0020	00040	SBLC	M8218	CPTX	SBLC WRITE DATA 06 H
05	0021	00041	SBLC	M8218	CPTX	SBLC WRITE DATA 07 H
05	0022	00042	SBLC	M8218	CPTX	SBLC WRITE DATA 08 H
05	0023	00043	SBLC	M8218	CPTX	SBLC WRITE DATA 09 H
05	0024	00044	SBLC	M8218	CPTX	SBLC WRITE DATA 10 H
05	0025	00045	SBLC	M8218	CPTX	SBLC WRITE DATA 11 H
05	0026	00046	SBLC	M8218	CPTX	SBLC WRITE DATA 12 H
05	0027	00047	SBLC	M8218	CPTX	SBLC WRITE DATA 13 H
05	0028	00050	SBLC	M8218	CPTX	SBLC WRITE DATA 14 H
05	0029	00051	SBLC	M8218	CPTX	SBLC WRITE DATA 15 H
05	002A	00052	SBLC	M8218	CPTX	TBMD EN SBI DATA L
05	002B	00053	SBLC	M8218	CPTX	BUS MD BYTE 0 PAR H

V-BUS DIRECTORY (CONT)

05	002C	00054	SBLC	M8218	CPTX	BUS MD BYTE 1 PAR H
05	002D	00055	SBLS	M8218	CPTX	SBLS SELECT SBI ADR L
05	002E	00056	SBLA	M8218	CPTX	ICLB IPL ACT 0 L
05	002F	00057	SBLA	M8218	CPTX	ICLB IPL ACT 1 L
05	0030	00060	SBHB	M8219	CPT3	SBHB WRITE DATA 16 H
05	0031	00061	SBHB	M8219	CPT3	SBHB WRITE DATA 17 H
05	0032	00062	SBHB	M8219	CPT3	SBHB WRITE DATA 18 H
05	0033	00063	SBHB	M8219	CPT3	SBHB WRITE DATA 19 H
05	0034	00064	SBHB	M8219	CPT3	SBHB WRITE DATA 20 H
05	0035	00065	SBHB	M8219	CPT3	SBHB WRITE DATA 21 H
05	0036	00066	SBHB	M8219	CPT3	SBHB WRITE DATA 22 H
05	0037	00067	SBHB	M8219	CPT3	SBHB WRITE DATA 23 H
05	0038	00070	SBHB	M8219	CPT3	SBHB WRITE DATA 24 H
05	0039	00071	SBHB	M8219	CPT3	SBHB WRITE DATA 25 H
05	003A	00072	SBHB	M8219	CPT3	SBHB WRITE DATA 26 H
05	003B	00073	SBHB	M8219	CPT3	SBHB WRITE DATA 27 H
05	003C	00074	SBHB	M8219	CPT3	SBHB WRITE DATA 28 H
05	003D	00075	SBHB	M8219	CPT3	SBHB WRITE DATA 29 H
05	003E	00076	SBHB	M8219	CPT3	SBHB WRITE DATA 30 H
05	003F	00077	SBHB	M8219	CPT3	SBHB WRITE DATA 31 H
05	0040	00100	SBHB	M8219	CPT1	SBHB RECEIVE MASK 0 H
05	0041	00101	SBHB	M8219	CPT1	SBHB RECEIVE MASK 1 H
05	0042	00102	SBHB	M8219	CPT1	SBHB RECEIVE MASK 2 H
05	0043	00103	SBHB	M8219	CPT1	SBHB RECEIVE MASK 3 H
05	0044	00104	SBHD	M8219	CPTX	BUS MD BYTE 2 PAR H
05	0045	00105	SBHD	M8219	CPTX	BUS MD BYTE 3 PAR H
05	0046	00106	SBHA	M8219	CPTX	SBHA BUFFER FULL L
05	0047	00107	SBHL	M8219	CPTX	SBLE LATE EXPECT RD L
05	0048	00110	SBHE	M8219	CPTX	SBLE REC PAR 0 H
05	0049	00111	SBHE	M8219	CPTX	SBLE REC PAR 1 H
05	004A	00112	SBHE	M8219	CPTX	SBLE REC PAR 2 H
05	004B	00113	SBHE	M8219	CPTX	SBLE REC PAR 3 H
05	004C	00114	SBHM	M8219	CPTX	TR SEL 1 L
05	004D	00115	SBHM	M8219	CPTX	TR SEL 2 L
05	004E	00116	SBHM	M8219	CPTX	TR SEL 4 L
05	004F	00117	SBHM	M8219	CPTX	TR SEL 8 L
05	0050	00120	SBHR	M8219	CPTX	TBMN BUF UMCT 0 L
05	0051	00121	SBHR	M8219	CPTX	TBMN BUF UMCT 1 L
05	0052	00122	SBHR	M8219	CPTX	TBMN BUF UMCT 2 L
05	0053	00123	SBHR	M8219	CPTX	TBMN BUF UMCT 3 L
05	0054	00124	SBHR	M8219	CPTX	TBMN BUF UADS L
05	0055	00125	SBHR	M8219	CPTX	TBMN BUF UFS L
05	0056	00126	SBHM	M8219	CPT0	SBHM SELECT SBI ADRS L
05	0057	00127	SBHR	M8219	CPTX	SBHR TRANS ENABLE L

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
05	0058	00130	SBMD	M8219	CPTX	TBMD EN SBI DATA L
05	0059	00131	SBME	M8219	CPTX	SBLE TRANS PAR L
05	005A	00132	SBMR	M8219	CPTX	SBLL TRANSMIT CA H
05	005B	00133	SBMM	M8219	CPTX	CEMM CUR MODE 0 H
05	005C	00134	SBMS	M8219	CPTX	CLKL SYS INIT B L
05	005D	00135	SBMM	M8219	CPTX	CEMM CUR MODE 1 H
05	005E	00136	SBMM	M8219	CPTX	TBMN DIS PROT L
05	005F	00137	SBMX	M8219	CPTX	RESERVED
05	0060	00140	SBMX	M8219	CPTX	RESERVED
05	0061	00141	SBMX	M8219	CPTX	RESERVED
05	0062	00142	SBMX	M8219	CPTX	RESERVED
05	0063	00143	SBMX	M8219	CPTX	RESERVED
05	0064	00144	SBMX	M8219	CPTX	RESERVED
05	0065	00145	SBMX	M8219	CPTX	RESERVED
05	0066	00146	SBMX	M8219	CPTX	RESERVED
05	0067	00147	SBMX	M8219	CPTX	RESERVED
05	0068	00150	SBMX	M8219	CPTX	RESERVED
05	0069	00151	SBMX	M8219	CPTX	RESERVED
05	006A	00152	SBMX	M8219	CPTX	RESERVED
05	006B	00153	SBMX	M8219	CPTX	RESERVED
05	006C	00154	SBMX	M8219	CPTX	RESERVED
05	006D	00155	SBMX	M8219	CPTX	RESERVED
05	006E	00156	SBMX	M8219	CPTX	RESERVED
05	006F	00157	SBMX	M8219	CPTX	RESERVED

V-BUS DIRECTORY (CONT)

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
06	0000	00000	FCTP	M8289	CPTX	DAPL ACC RA CONTEXT 0 H
06	0001	00001	FCTP	M8289	CPTX	DAPL ACC RA CONTEXT 1 H
06	0002	00002	FCTC	M8289	CPTX	FCTC CLR RR L
06	0003	00003	FCTH	M8289	CPTX	FCTH CP SYNC H
06	0004	00004	FNME	M8289	CPTX	FNME BUS_EXP L
06	0005	00005	FCTJ	M8289	CPTX	FCTJ ACC N DATA H
06	0006	00006	FCTC	M8289	CPTX	FCTC ACC Z DATA H
06	0007	00007	FCTC	M8289	CPTX	FCTC ACC V DATA H
06	0008	00010	FNMT	M8289	CPT3	FCTD RA ADRS 3 L
06	0009	00011	FNMT	M8289	CPT3	FCTD RA ADRS 2 L
06	000A	00012	FNMT	M8289	CPT3	FCTD RA ADRS 1 L
06	000B	00013	FNMT	M8289	CPT3	FCTD RA ADRS 0 L
06	000C	00014	FNMT	M8289	CPT3	FCTP RB ADRS 3 L
06	000D	00015	FNMT	M8289	CPT3	FCTP RB ADRS 2 L
06	000E	00016	FNMT	M8289	CPT3	FCTP RB ADRS 1 L
06	000F	00017	FNMT	M8289	CPT3	FCTP RB ADRS 0 L
06	0010	00020	XXXX	M8289	CPTX	RESERVED
06	0011	00021	XXXX	M8289	CPTX	RESERVED
06	0012	00022	FNMT	M8289	CPTX	EALU C 0 L
06	0013	00023	FNMT	M8289	CPTX	FCTE COMPL L
06	0014	00024	FNMT	M8289	CPTX	FADA SPC (0) H
06	0015	00025	FNMT	M8289	CPTX	FNMS EALU CIN L
06	0016	00026	FNMT	M8289	CPTX	FCTC SEL NORM H
06	0017	00027	FNMT	M8289	CPTX	RESERVED
06	0018	00030	FNMT	M8288	CPT2	FCTN LOAD AR0 H
06	0019	00031	FNMT	M8288	CPT2	FCTN LOAD AR1 H
06	001A	00032	FNMT	M8288	CPT2	FCTN LOAD ARX H
06	001B	00033	FNMT	M8288	CPT2	FCTN LOAD BR1 H
06	001C	00034	FNMT	M8288	CPT2	FCTN LOAD BR0 H
06	001D	00035	FNMT	M8288	CPT0	FADS BUS_FAD L
06	001E	00036	FNMT	M8288	CPTX	RESERVED
06	001F	00037	FNMT	M8288	CPTX	RESERVED
06	0020	00040	FNMT	M8288	CPT1	FCTN FAMX EN 0 L
06	0021	00041	FNMT	M8288	CPT3	FCTA A GT B H
06	0022	00042	FNMT	M8288	CPT3	FCTN SHF MUX EN1 L
06	0023	00043	FNMT	M8288	CPT3	FCTN SHF MUX EN0 L
06	0024	00044	FNMT	M8288	CPT1	FCTN FALU FUNC SEL 2 H
06	0025	00045	FNMT	M8288	CPT1	FCTN FALU FUNC SEL 1 H
06	0026	00046	FNMT	M8288	CPT1	FCTN FALU FUNC SEL 0 H
06	0027	00047	FNMT	M8288	CPT1	FCTN FAMX SEL 1 H

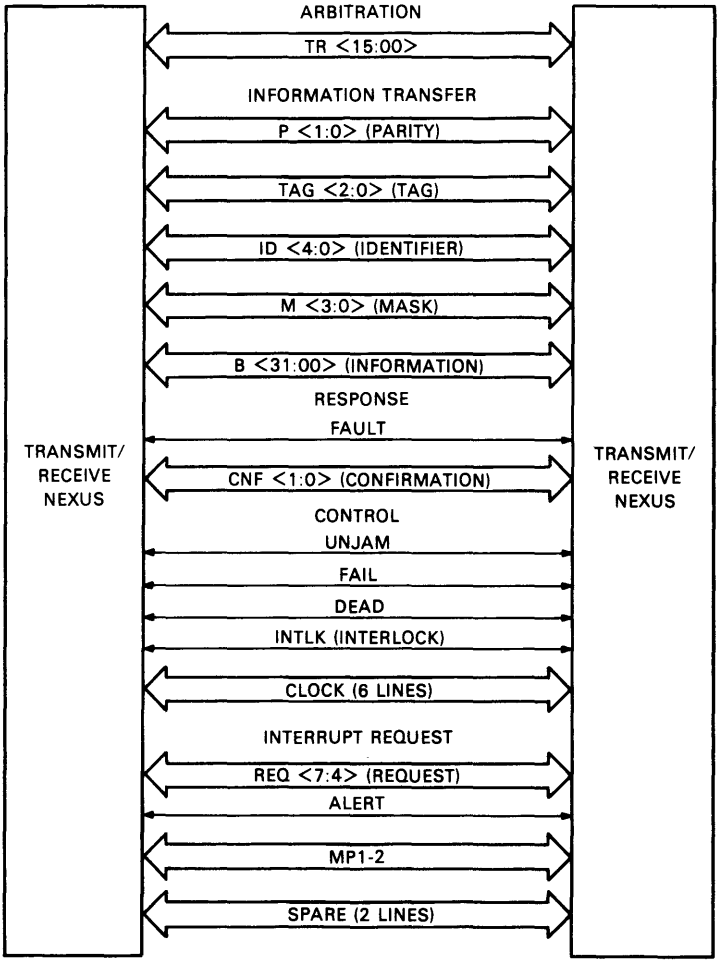
V-BUS DIRECTORY (CONT)

06	0028	00050	FNMT	M8228	CPT3	FCTF SHF COUNT 5 H
06	0029	00051	FNMT	M8228	CPT3	FCTF SHF COUNT 4 H
06	002A	00052	FNMT	M8228	CPT3	FCTF SHF COUNT 3 H
06	002B	00053	FNMT	M8228	CPT3	FCTF SHF COUNT 2 H
06	002C	00054	FNMT	M8228	CPT3	FCTF SHF COUNT 1 H
06	002D	00055	FNMT	M8228	CPT3	FCTF SHF COUNT 0 H
06	002E	00056	FNMT	M8228	CPT3	FCTN FALU CARRY IN H
06	002F	00057	FNMT	M8228	CPT1	FCTN FAMX SEL 0 H

CHAPTER 6

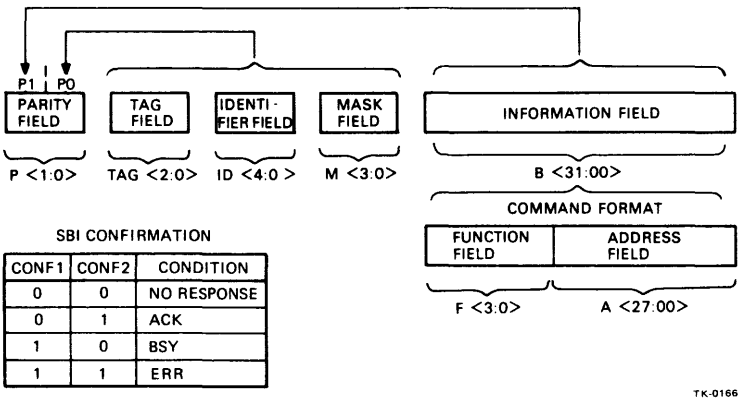
SYSTEM BACKPLANE INTERCONNECT

SBI CONFIGURATION



TK-0077

SBI PARITY FIELD CONFIGURATION

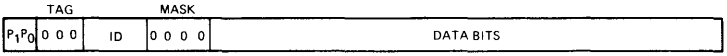


SBI CONFIRMATION

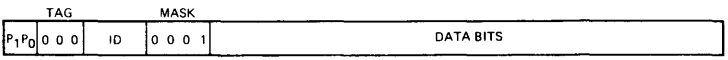
CONF1	CONF2	CONDITION
0	0	NO RESPONSE
0	1	ACK
1	0	BSY
1	1	ERR

SBI INFORMATION TRANSFER FORMATS

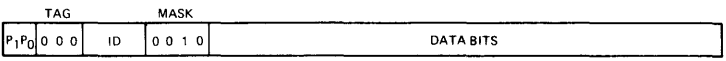
READ DATA FORMAT



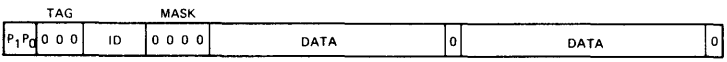
CORRECTED READ DATA FORMAT



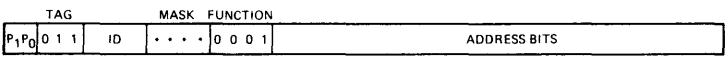
READ DATA SUBSTITUTE FORMAT



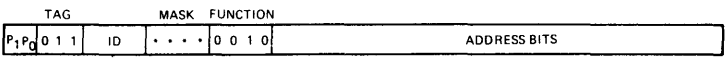
INTERRUPT SUMMARY RESPONSE FORMAT



COMMAND ADDRESS FORMAT FOR READ MASKED



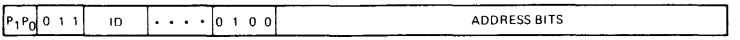
COMMAND ADDRESS FORMAT FOR WRITE MASKED



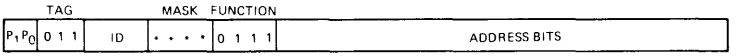
TK-0723

SBI INFORMATION TRANSFER FORMATS (CONT)

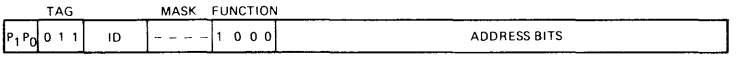
COMMAND ADDRESS FORMAT FOR INTERLOCK READ MASKED



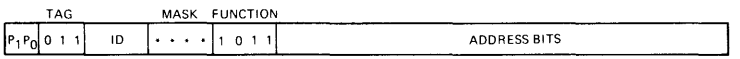
COMMAND ADDRESS FORMAT FOR INTERLOCK WRITE MASKED



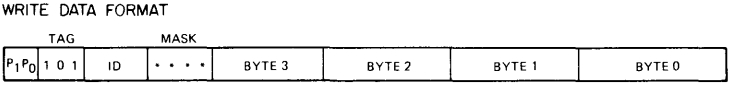
COMMAND ADDRESS FORMAT FOR EXTENDED READ



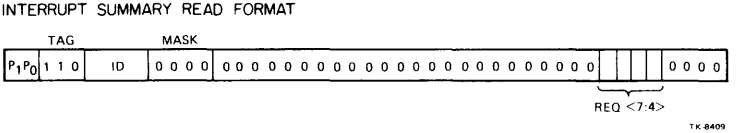
COMMAND ADDRESS FORMAT FOR EXTENDED WRITE MASKED



WRITE DATA FORMAT



INTERRUPT SUMMARY READ FORMAT



TK-8409

SBI FIELD DESCRIPTION

Field	Description
Arbitration Group	
Arbitration Field [TR (15:00)]	Establishes a fixed priority among nexus for access to and control of the information transfer path.
Information Transfer Group	
Information Field [B (31:00)]	Bidirectional lines that transfer data, command/address, and interrupt information between nexus.
Mask Field [M (3:00)D]	Primary function: encoded to indicate a particular byte within the 32-bit information field [B (31:00)].
	Secondary function: in conjunction with the tag field, indicates a particular type of read data.
Identifier Field [ID (4:0)]	Identifies the logical source or destination of information contained in B (31:00).
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 (1:0)]	Provides even parity for all information transfer path fields.
Response Group	
Confirmation Field [CNF (1:0)]	Encoded by a receiving nexus to specify one of four response types and indicate its capability to respond to the transmitter's request.
Fault Field (FAULT)	A cumulative error line to the CPU that 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 (CONT)

Field	Description
Interrupt Request Group	
Request Field [REQ (7:4)]	Allows a nexus to request an interrupt to service a condition requiring CPU intervention. Each request line represents a level of nexus request priority.
Alert Field (ALERT)	A cumulative status line that allows those nexus 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.
Fail Field (FAIL)	A single line from the restart nexus to provide a restart signal to the CPU to initiate a system restart operation.
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 attached nexus that initiates a restore operation.
Interlock Field (INTLK)	A single line that provides coordination among nexus responding to certain read/write commands to ensure exclusive access to shared data structures.

CONSOLE (PHYSICAL 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

SBI ADDRESS 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 X X

	1	0 0 0 0 0 0 0 0 0 0 0 0	TR#	REGISTER ADDRESS	

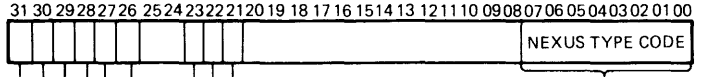
TR# (Base 10)	Physical Address (Hex)	SBI Address (Hex)
0	20000000	8000000
1	20002000	8000800
2	20004000	8001000
3	20006000	8001800
4	20008000	8002000
5	2000A000	8002800
6	2000C000	8003000
7	2000E000	8003800
8	20010000	8004000
9	20012000	8004800
10	20014000	8005000
11	20016000	8005800
12	20018000	8006000
13	2001A000	8006800
14	2001C000	8007000
15	2001E000	8007800

SBI Device Vector Generation

8	7 6	5 4 3 2	1	0
1	REQ/BR LEVEL	TR NO.	0	0

LEV 4 = 0 0
LEV 5 = 0 1
LEV 6 = 1 0
LEV 7 = 1 1

SBI FAULTS (ADAPTER CONFIGURATION REGISTER)



PAR FLT
URD FLT
MXT FLT
WSQ FLT
ISQ FLT
XMT FLT
PWR DWN
PWR UP
OVR TMP

0 0 0 X X X X X	MEMORY	
0 0 1 0 0 0 0 0	MBA	
0 0 1 0 1 0 0 0	UBA	0
0 0 1 0 1 0 0 1		1
0 0 1 0 1 0 1 0		2
0 0 1 0 1 0 1 1		3
0 1 0 0 0 0 Y Y*	MA780	
0 0 1 1 0 0 0 0	DR780	

*YY = PORT NUMBER

31 — PARITY FAULT
30 — WRITE SEQUENCE FAULT
29 — UNEXPECTED READ DATA FAULT
28 — INTERLOCK SEQUENCE FAULT
27 — MULTIPLE TRANSMITTER FAULT
26 — TRANSMITTER DURING CYCLE THAT CAUSED FAULT
23 — POWER DOWN
22 — POWER UP
21 — OVER TEMPERATURE

TK-0695

SBI CONFIGURATION RULES FOR TR SELECTION

The following rules are suggested for selecting TR levels for NEXUS on the SBI. They apply to VAX-11/780 and VAX-11/782 systems. These rules are guidelines; deviation may be desirable and, in some cases, necessary.

1. The TR level of a device determines its relative priority in competing for SBI access. High TR levels mean low priority. TR 1 is the highest priority and TR 15 the lowest to which any NEXUS (other than the CPU) may be assigned.

2. The standard order of assigning devices to TR levels is:

   - All MS780s
   - All MA780-Cs
   - All DW780s
   - All RH780s
   - DR780
   - CI780
   - CPU

   The default TR assignments currently used by manufacturing are:

First MS780	= TR 1
First MA780 port	= TR 2
First DW780	= TR 3
First RH780 (for disks)	= TR 8
Second RH780 (for tapes)	= TR 9
DR780	= TR 13
CI780	= TR 14
CPU	= TR 16 (the implied TR level)

   These defaults should cover most VAX-11/780 systems. Large systems, as well as some applications, will require field changes to TR levels.

   When selecting TR levels for complex systems, you should try to minimize latency effects (such as data lates) in memory access. Therefore, always give memory controllers the lowest TR levels and assign the CPU to TR 16. RH780s and DW780s for peripheral devices should be assigned TR levels based on the sensitivity of the devices (or device controllers) to memory latency.

3. One to two MS780s per system. Interleaving with MS780s requires consecutive TR levels as well as equal amounts of memory in the even/odd pair; consecutive TR levels are strongly suggested even without interleaving. When a MS780-A and MS780-C controller reside on the same system, the MS780-C should be assigned the lowest TR level.

4. Zero to four MA780-Cs per system. TR levels must be consecutive. A given MA780 must have the same TR on all SBIs. MS780 memory controllers should always have lower TRs than MA780-C memory ports.

5. One to four DW780s per system.

6. Zero to four RH780s per system. In general, RH780s used for disks should have higher priority than RH780s used only for tapes. The main criterion for allocating devices to RH780s and RH780s to TRs is the relative sensitivity of the device to memory latency. Since the RH780 has less than one disk sector of buffering, the memory latency requirement is determined (slightly pessimistically) by the device data rate:

RP07 at 2.2 Mbyte/sec	3.64 microsec/quadword
RP07 at 1.3	6.15 microsec/quadword
RM03, RM05, RM80	6.67 microsec/quadword
RP04/5/6	10.00 microsec/quadword
TU78 at 6250 BPI	10.24 microsec/quadword
TU77 at 1600 BPI	40.00 microsec/quadword
TE16 at 1600 BPI	>100.00 microsec/quadword

These figures define the average response time requirement as seen from the device. Due to RH780 buffering, RH780 devices can withstand an occasional memory response time of nearly three times the average figure just given without causing a data late.

The RP07 at 2.2 megabytes per second requirement is high enough that only interleaved MS780 memory can be used. It is not expected to function on a VAX-11/782 or a VAX-11/780 with MA780 unless the application prevents RP07 I/O from accessing the MA780 address space.

In determining the TR level for RH780s with more than one type of device, only the memory response time requirement of the fastest device on the RH780s needs to be considered.

7. Zero to one DR780 per system. It is suggested that DR780 pairs that are used to interconnect VAX-11/780 systems use the same TR assignment in each VAX-11/780.

The data rate of the DR780 may be set as low as 156 kilobytes per second or as high as 8 megabytes per second depending on the attributes of the connected device. 156 kilobytes per second is slower than a TU77, while 8 megabytes per second is faster than any current VAX-11/780 memory will support.

Since the DR780 is synchronous and not fully buffered, it may experience data lates. The only solutions are faster (interleaved) memory, prevention of concurrent I/O from other devices on the SBI (usually not practical), and decreasing the DR780 data rate setting.

8. One CI780 per system. It is suggested that for a CI network, the same TR level be assigned to each CI780.

The CI780 is synchronous and fast (8.75 megabytes per second peak), but it is fully buffered. Therefore, CI780 should have the highest TR number (lowest priority) of any device other than the CPU.

9. One CPU. Only one CPU per SBI is supported, and it must be at (implied) TR 16.

SBI CONFIGURATION RULES FOR TR SELECTION (CONT)

10. These TR selection rules are mainly useful when configuring VAX-11/780 and VAX-11/782 systems with many fast peripheral devices. On systems with little I/O activity, the SBI and memory will be so lightly loaded that TR selection has little importance. On the other hand, when the memory demand greatly exceeds the capacity of the SBI and the memory, TR selection will not make the memory cycle faster.

    TR selection is important for any system with aggregate bandwidth of concurrently active DMA devices above two megabytes per second.

11. If you run out of TR levels, you need another VAX-11/780.

12. The following example illustrates these guidelines for a large VAX-11/780 system consisting of: two MS780-Cs, two MA780-Cs, two DR780s, three RH780s, and one CI780.

Device	TR
MS780-C	1
MS780-C	2
MA780-C	3
MA780-C	4
DW780	5
DW780	6
RH780	8
RH780	9
RH780	10
CI780	14

	A	B	C	D	E	F
A1				BUS SBI DEAD L		BUS SBI INTLK
A2	+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V
B1	BUS SBI B00 L		BUS SBI B20 L	BUS SBI M0 L	BUS SBI REQ4 L	BUS SBI TR00
B2						
C1	BUS SBI B01 L		BUS SBI B21 L	BUS SBI M1 L	BUS SBI REQ5 L	BUS SBI TR01
C2	GND		GND	GND	GND	GND
D1	BUS SBI B02 L		BUS SBI B22 L	BUS SBI M2 L	BUS SBI REQ6 L	BUS SBI TR02
D2			BUS SBI B23 L	BUS SBI M3 L	BUS SBI REQ7 L	+12 V
E1	BUS SBI B03 L					BUS SBI TR03
E2						
F1					BUS SBI TP H	
F2		BUS SBI B12 L		BUS SBI P0 L	BUS SBI TP L	BUS SBI TR04
H1	GND	GND	GND	GND	GND	GND
H2		BUS SBI B13 L		BUS SBI P1 L	BUS SBI PCLK H	BUS SBI TR05
J1		BUS SBI B14 L		BUS SBI SPARE 0	BUS SBI PCLK L	BUS SBP TP06
J2		BUS SBI B15 L		BUS SBI SPARE 1	BUS SBI PDCLK H	BUS SBI TR07
K1					- 5 V	
K2					BUS SBI PDCLK L	
L1						
L2		- 5 V				
M1	BUS SBI B04 L		BUS SBI B24 L	BUS SBI TAG0 L	BUS SBI MP1 L	BUS SBI TR08
M2						
N1	BUS SBI B05 L		BUS SBI B25 L	BUS SBI TAG1 L	BUS SBI MP2 L	BUS SBI TR09
N2	GND	GND	GND	GND	GND	GND
P1	BUS SBI B06 L		BUS SBI B26 L	BUS SBI TAG2 L	BUS SBI UNJAM L	BUS SBI TR10
P2	BUS SBI B07 L		BUS SBI B27 L	BUS SBI ID0 L	BUS SBI ALERT L	BUS SBI TR11
R1						
R2						
S1	+12 V					
S2	BUS SBI B08 L	BUS SBI B16 L	BUS SBI B28 L	BUS SBI ID1 L	BUS SBI CNF0 L	BUS SBI TR12
T1	GND	GND	GND	GND	GND	GND
T2	BUS SBI B09 L	BUS SBI B17 L	BUS SBI B29 L	BUS SBI ID2 L	BUS SBI CNF1 L	BUS SBI TR13
U1	BUS SBI B10 L	BUS SBI B18 L	BUS SBI B30 L	BUS SBI ID3 L	BUS SBI FAULT L	BUS SBI TR14
U2	BUS SBI B11 L	BUS SBI B19 L	BUS SBI B31 L	BUS SBI ID4 L		BUS SBI TR15
V1	+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V	+ 5 V
V2			BUS SBI FAIL L			

CHAPTER 7
SBI NEXUS

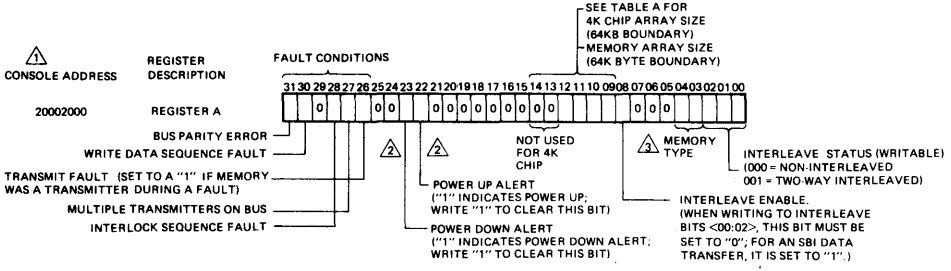


TABLE A 4K CHIP

12	11	10	09	MEMORY ARRAY SIZE
0	0	0	0	64K BYTE
0	0	0	1	128K BYTE
0	0	1	0	192K BYTE
0	0	1	1	256K BYTE
0	1	0	0	320K BYTE
0	1	0	1	384K BYTE
0	1	1	0	448K BYTE
0	1	1	1	512K BYTE
1	0	0	0	576K BYTE
1	0	0	1	640K BYTE
1	0	1	0	704K BYTE
1	0	1	1	768K BYTE
1	1	0	0	832K BYTE
1	1	0	1	896K BYTE
1	1	1	0	960K BYTE
1	1	1	1	1024K BYTE

TABLE B 16K CHIP

14	13	12	11	10	09	MEMORY ARRAY SIZE
0	0	0	0	–	–	256K BYTE
0	0	0	1	–	–	512K BYTE
0	0	1	0	–	–	768K BYTE
0	0	1	1	–	–	1024K BYTE
0	1	0	0	–	–	1280K BYTE
0	1	0	1	–	–	1536K BYTE
0	1	1	0	–	–	1792K BYTE
0	1	1	1	–	–	2048K BYTE
1	0	0	0	–	–	2304K BYTE
1	0	0	1	–	–	2560K BYTE
1	0	1	0	–	–	2816K BYTE
1	0	1	1	–	–	3072K BYTE
1	1	0	0	–	–	3328K BYTE
1	1	0	1	–	–	3584K BYTE
1	1	1	0	–	–	3840K BYTE
1	1	1	1	–	–	4096K BYTE

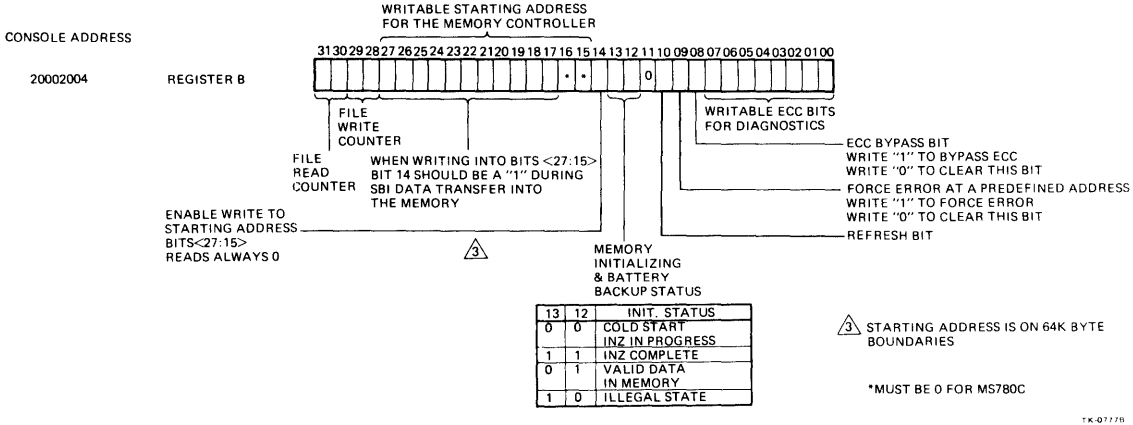
1 SINGLE CONTROLLER WITH TR LEVEL = 1

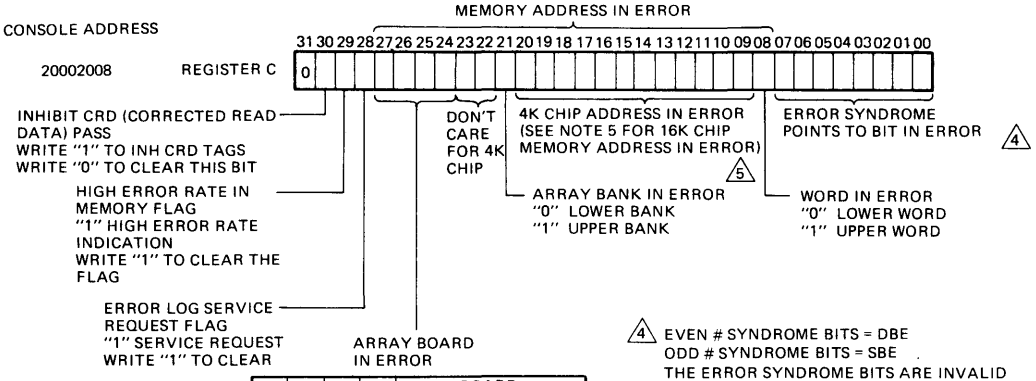
2 THESE BITS ARE USED ONLY BY MEMORY CONTROLLERS NOT HAVING A ROM BOOTSTRAP

3

MEMORY TYPE		
0	0	ERROR CONDITION; NO ARRAY BOARDS PLUGGED INTO BACKPLANE
0	1	4K MOS
1	0	16K MOS
1	1	ERROR CONDITION; BOTH 4K AND 16K BOARDS PLUGGED INTO BACKPLANE

TK-0777C


WRITABLE STARTING ADDRESS FOR THE MEMORY CONTROLLER
CONSOLE ADDRESS
20002004
REGISTER B
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
0
FILE WRITE COUNTER
FILE READ COUNTER
WHEN WRITING INTO BITS <27:15> BIT 14 SHOULD BE A "1" DURING SBI DATA TRANSFER INTO THE MEMORY
ENABLE WRITE TO STARTING ADDRESS BITS<27:15> READS ALWAYS 0
3
WRITABLE ECC BITS FOR DIAGNOSTICS
ECC BYPASS BIT
WRITE "1" TO BYPASS ECC
WRITE "0" TO CLEAR THIS BIT
FORCE ERROR AT A PREDEFINED ADDRESS
WRITE "1" TO FORCE ERROR
WRITE "0" TO CLEAR THIS BIT
REFRESH BIT
MEMORY INITIALIZING & BATTERY BACKUP STATUS
13 12 INIT. STATUS
0 0 COLD START INZ IN PROGRESS
1 1 INZ COMPLETE
0 1 VALID DATA IN MEMORY
1 0 ILLEGAL STATE
3 STARTING ADDRESS IS ON 64K BYTE BOUNDARIES
*MUST BE 0 FOR MS780C
TK-07776



27	26	25	24	BOARD
0	0	0	0	BOARD 1
0	0	0	1	BOARD 2
0	0	1	0	BOARD 3
0	0	1	1	BOARD 4
0	1	0	0	BOARD 5
0	1	0	1	BOARD 6
0	1	1	0	BOARD 7
0	1	1	1	BOARD 8
1	0	0	0	BOARD 9
1	0	0	1	BOARD 10
1	0	1	0	BOARD 11
1	0	1	1	BOARD 12
1	1	0	0	BOARD 13
1	1	0	1	BOARD 14
1	1	1	0	BOARD 15
1	1	1	1	BOARD 16

4 EVEN # SYNDROME BITS = DBE
ODD # SYNDROME BITS = SBE
THE ERROR SYNDROME BITS ARE INVALID UNLESS THE ERROR LOG SERVICE REQUEST BIT (BIT 28, REG 0) IS SET TO A "1".

5 REGISTER C, BITS <22:09> ARE 16K CHIP ADDRESS IN ERROR.
BIT <23> IS THE ARRAY BANK IN ERROR (0 IS LOWER BANK & 1 IS THE UPPER BANK)
THE MEANING FOR ALL OTHER BITS IN REGISTER C IS SAME FOR 4K & 16K CHIPS.

NOTE: WHEN INSTALLING AN MS780A & MS780C ON THE SAME 11/780, THE MS780C SHOULD HAVE THE LOWEST TR.

TK-0777A

MEMORY ARRAY ADDRESSES

Array	M8211 Size	Address Range	M8210 Size	Address Range
1	64 K	0- FFFF	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

b	d	f	j	l	n	r	t	v	x	z	bb	dd	ff	jj	ll	nn	rr	tt	vv	MS780 Configuration
a	c	e	h	k	m	p	s	u	w	y	aa	cc	ee	hh	kk	mm	pp	ss	uu	for REV A Backpanel
W6	W7							W8								W9	W10	W11	W12	

*	*	*	*	*			*	*			*			*	*	*	*			Configuration for
*	*	*	*	*	SA1		*	*	0	IRD	*	TR8		*	*	*	*	TR 1		REV -- Backpanel
W1	W2	W3	W4	W5			W6	W7			W8			W9	W10	W11	W12			

TR Arbitration Level

Signal Name	TR SEL 8	TR SEL 4	TR SEL 2	TR SEL 1	Wire Wrap F20 K2 to
TR#	W9	W10	W11	W12	
1	--	--	--	-- [1]	F20 C1
2	--	--	--	I [2]	F20 D1
3	--	--	I	--	F20 E1
4	--	--	I	I	F20 F2
5	--	I	--	--	F20 H2
6	--	I	--	I	F20 J1
7	--	I	I	--	F20 J2
8	--	I	I	I	F20 M1
9	I	--	--	--	F20 N1
10	I	--	--	I	F20 P1
11	I	--	I	--	F20 P2
12	I	--	I	I	F20 S2
13	I	I	--	--	F20 T2
14	I	I	--	I	F20 U1
15	I	I	I	--	F20 U2

Inhibit ROM Decode

W8=I	No response[2] confirmation for Read to ROM Address Space	NOTES: 1. When installing an MS780A and MS780C on the same VAX-11/780 system, the MS780C should be the first memory (i.e., have the lowest TR). 2. The memory with the ROM bootstrap must be at TR1.
W8=--	Read to ROM[1] Address Space receive Normal confirmation	

Starting Address

	W6	W7 [1] [2]
0	--	--
4 Mega Byte	--	I
8 Mega Byte	I	--
12 Mega Byte	I	I

W1-W5 Spare

[1] - Standard for Memory 1
[2] - Standard for Memory 2

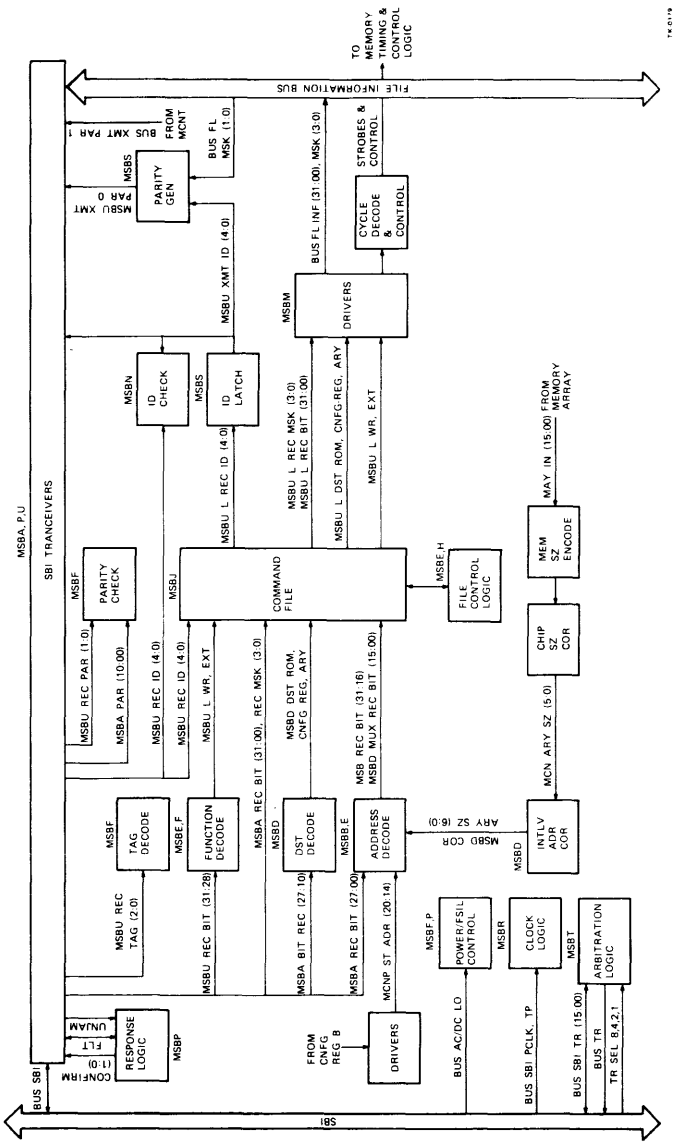
20	M8214 MSB	
19	M8213 MCN	
18	M8212 MDT	
17	M8211 MAY	
16	M8211 MAY	
15	M8211 MAY	*
14	M8211 MAY	*
13	M8211 MAY	*
12	M8211 MAY	*
11	M8211 MAY	*
10	M8211 MAY	*
9	M8211 MAY	*
8	M8211 MAY	*
7	M8211 MAY	*
6	M8211 MAY	*
5	M8211 MAY	*
4	M8211 MAY	*
3	M8211 MAY	*
2	M8211 MAY	*
1	M9040 TRM	*

* When not installed, use blank module 7014103.

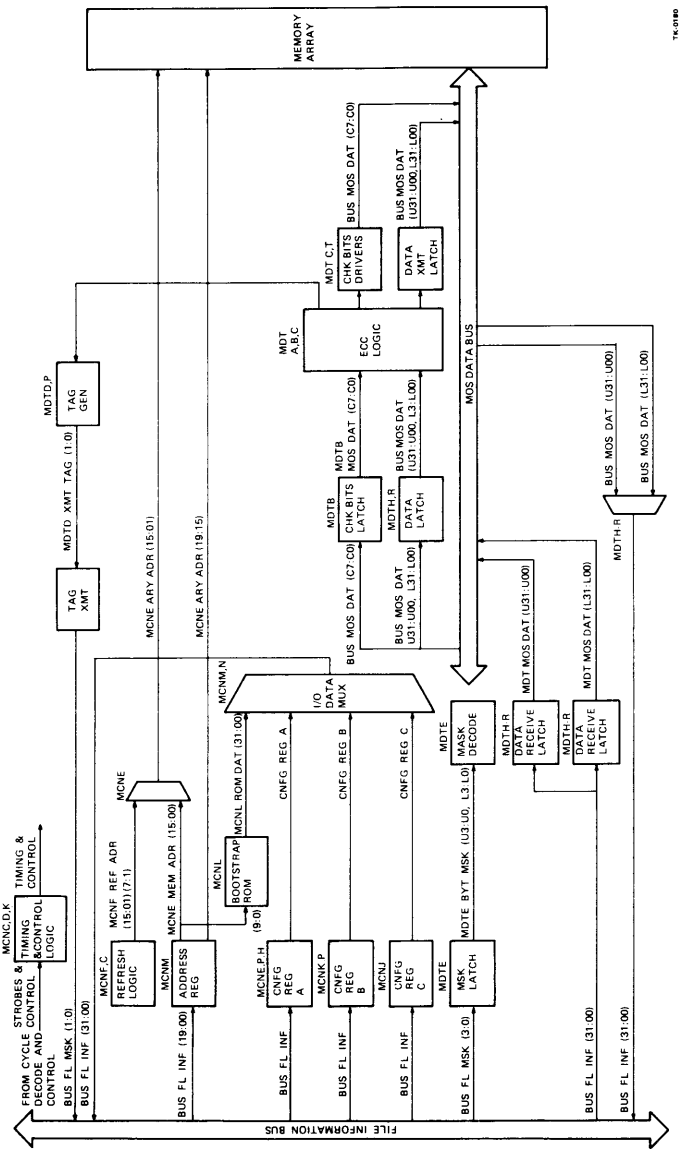
20	M8214 MSB	
19	M8213 MCN	
18	M8212 MDT	
17	M8210 MAY	
16	M8210 MAY	
15	M8210 MAY	*
14	M8210 MAY	*
13	M8210 MAY	*
12	M8210 MAY	*
11	M8210 MAY	*
10	M8210 MAY	*
9	M8210 MAY	*
8	M8210 MAY	*
7	M8210 MAY	*
6	M8210 MAY	*
5	M8210 MAY	*
4	M8210 MAY	*
3	M8210 MAY	*
2	M8210 MAY	*
1	M9040 TRM	*

* When not installed, use blank module 7014103.

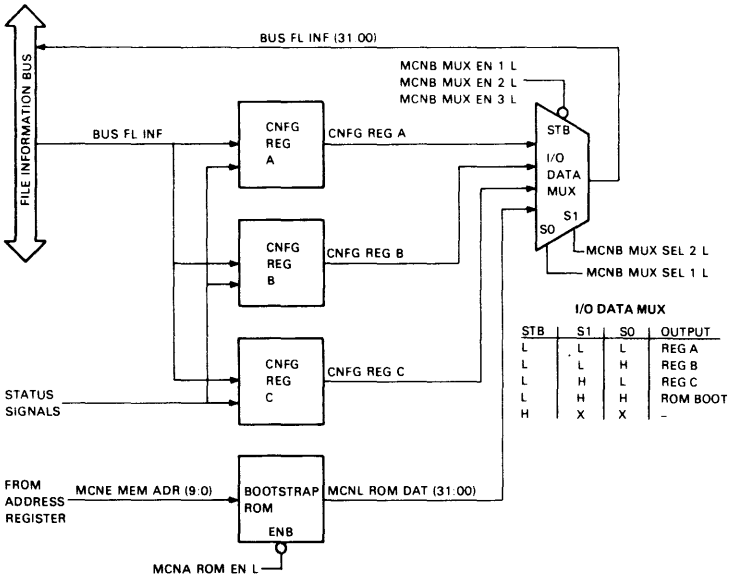
MEMORY BLOCK DIAGRAM, PART 1



MEMORY BLOCK DIAGRAM, PART 2

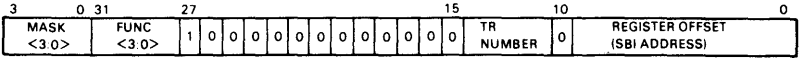


MEMORY I/O DATA LOGIC



UBA ADDRESS SPACE AND C/A FORMAT

SBI C/A Format for UBA Register Access



TK-032d

Base Addresses

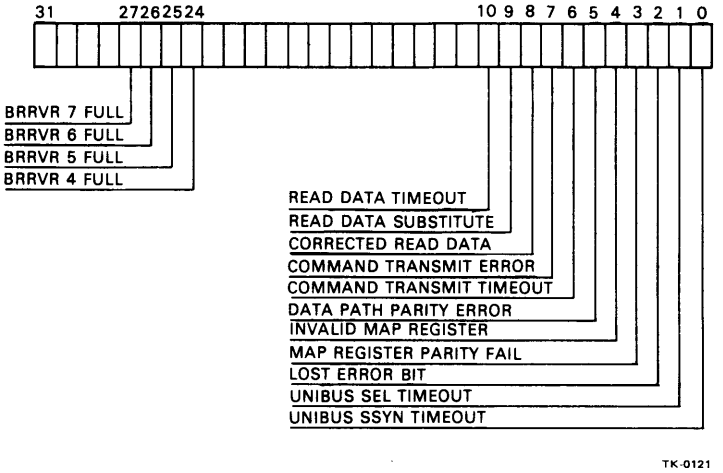
TR Num Base 10	Base Address (Physical hex)	SBI Address (hex)	TR Num Base 10	Base Address (Physical hex)	SBI Address (hex)
			8	20010000	8004000
1	20002000	8000800	9	20012000	8004800
2	20004000	8001000	10	20014000	8005000
3	20006000	8001800	11	20016000	8005800
4	20008000	8002000	12	20018000	8006000
5	2000A000	8002800	13	2001A000	8006800
6	2000C000	8003000	14	2001C000	8007000
7	2000E000	8003800	15	2001E000	8007800

Register Offsets

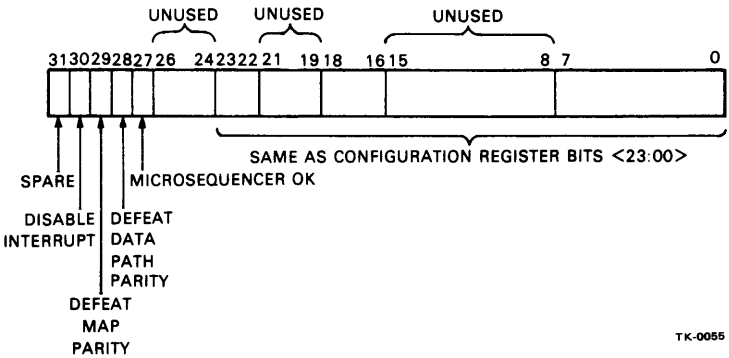
UBA Reg	Byte Address (Physical hex)	SBI Address (hex)	UBA Reg	Byte Address (Physical hex)	SBI Address (hex)
CNFGR	000	000	.	.	.
UBACR	004	001	.	.	.
UBASR	008	002	DPR 14	078	01E
DCR	00C	003	DPR 15	07C	01F
FMER	010	004	Reserved	080	020
FUBAR	014	005	.	.	.
FMER	018	006	.	.	.
FUBAR	01C	007	Reserved	7EC	1FF
BRSVR 0	020	008	MR 0	800	200
BRSVR 1	024	009	MR 1	804	201
BRSVR 2	028	00A	.	.	.
BRSVR 3	02C	00B	.	.	.
BRRVR 4	030	00C	MR 494	FB8	3EE
BRRVR 5	034	00D	MR 495	FBC	3EF
BRRVR 6	038	00E	Reserved	FC0	3F0
BRRVR 7	03C	00F	.	.	.
DPR 0	040	010	Reserved	FFC	3FF
DPR 1	044	011			

UBA REGISTERS

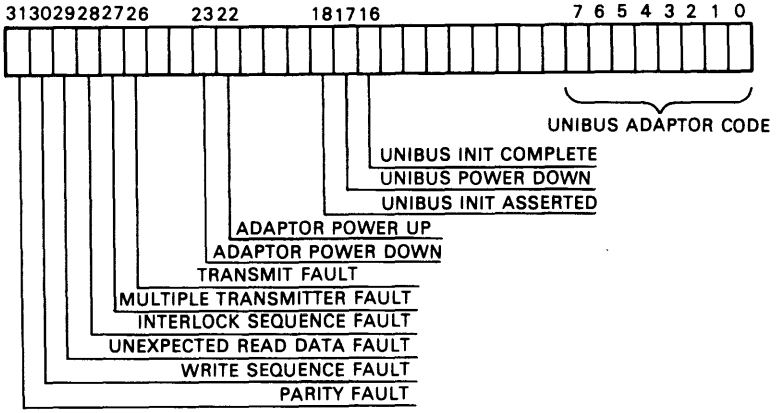
UBA STATUS REGISTER, BIT CONFIGURATION



UBA DIAGNOSTIC CONTROL REGISTER, BIT CONFIGURATION

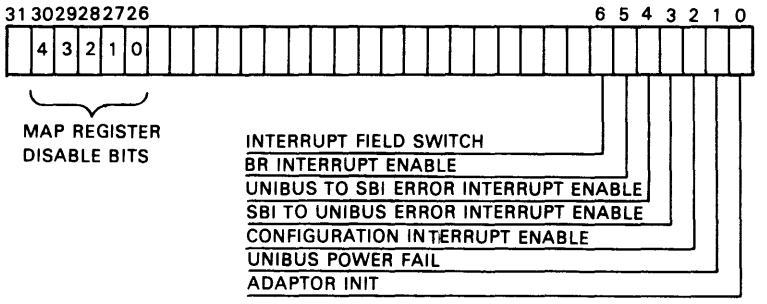


UBA CONFIGURATION REGISTER, BIT CONFIGURATION



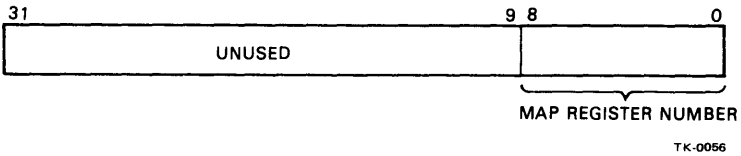
TK-0119

UBA CONTROL REGISTER, BIT CONFIGURATION

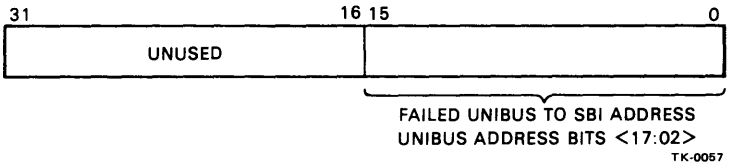


TK-0120

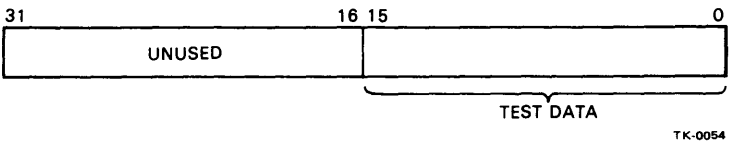
UBA FAILED MAP ENTRY REGISTER, BIT CONFIGURATION



UBA FAILED UNIBUS ADDRESS REGISTER, BIT CONFIGURATION

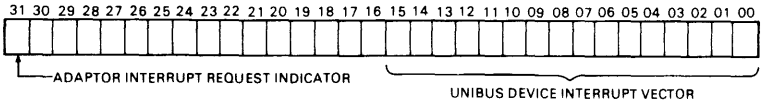


UBA BUFFER SELECTION VERIFICATION REGISTER, BIT CONFIGURATION

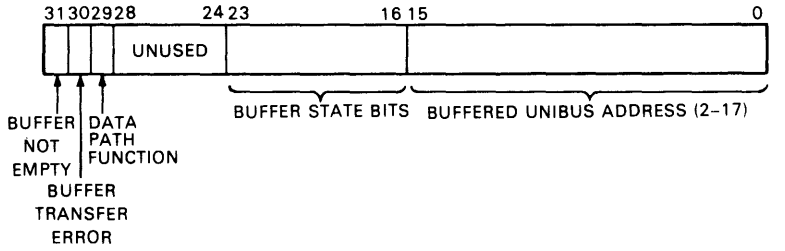


NOTE:
THE INFORMATION FOUND IN THESE REGISTERS IS MEANINGFUL ONLY IF A CORRESPONDING ERROR BIT IS FOUND IN THE UBA STATUS REGISTER.

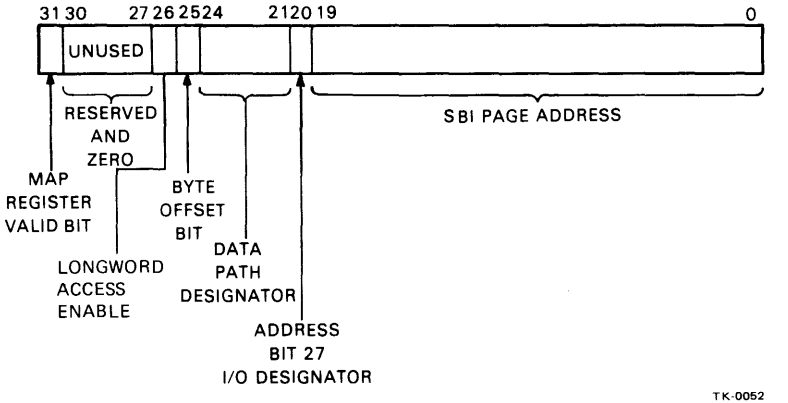
UBA BR RECEIVE VECTOR REGISTER, BIT CONFIGURATION



UBA DATA PATH REGISTER, BIT CONFIGURATION



UBA MAP REGISTER, BIT CONFIGURATION



b	d	f	j	l	n	r	t	v	x	z	bb	dd	ff	jj	ll	nn	rr	tt	vv	DW780 Configuration
a	c	e	h	k	m	p	s	u	w	y	aa	cc	ee	hh	kk	mm	pp	ss	uu	for REV A Backpanel
W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15						
*	*	*	*	*	*	*	*	*	*	*	*	*	*							Configuration for
*	*	*	*	*	*	*	*	*	*	*	*	*	*							REV -- Backpanel
W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14							

TR Arbitration Level

Signal Name	USIC TR SEL A L	USIC TR SEL B L	USIC TR SEL C L	USIC TR SEL D L	Wire Wrap DØ1R2 to
TR#	W3	W4	W5	W6	
1	--	--	--	--	FØ1C1
2	I	--	--	--	FØ1D1
3	--	I	--	-- *	FØ1E1
4	I	I	--	--	FØ1F2
5	--	--	I	--	FØ1H2
6	I	--	I	--	FØ1J1
7	--	I	I	--	FØ1J2
8	I	I	I	--	FØ1M1
9	--	--	--	I	FØ1N1
10	I	--	--	I	FØ1P1
11	--	I	--	I	FØ1P2
12	I	I	--	I	FØ1S2
13	--	--	I	I	FØ1T2
14	I	--	I	I	FØ1U1
15	--	I	I	I	FØ1U2

* Normal for first DW780

UNIBUS Address Space Select

Signal Name	USID Adapter Ø L	USID Adapter 1 L
Adapter#	W2	W1
Ø	--	-- *
1	I	--
2	--	I
3	I	I

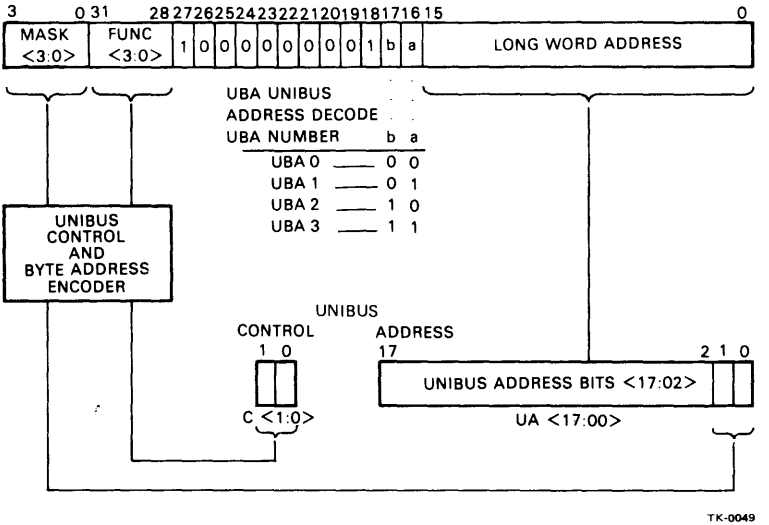
Interrupt Level Selection Signal Name

	UAIF SBI PRI JMP Ø L	UAIF SBI PRI JMP 1 L
ISR#	W7	W8
4	--	-- *
5	I	--
6	--	I
7	I	I

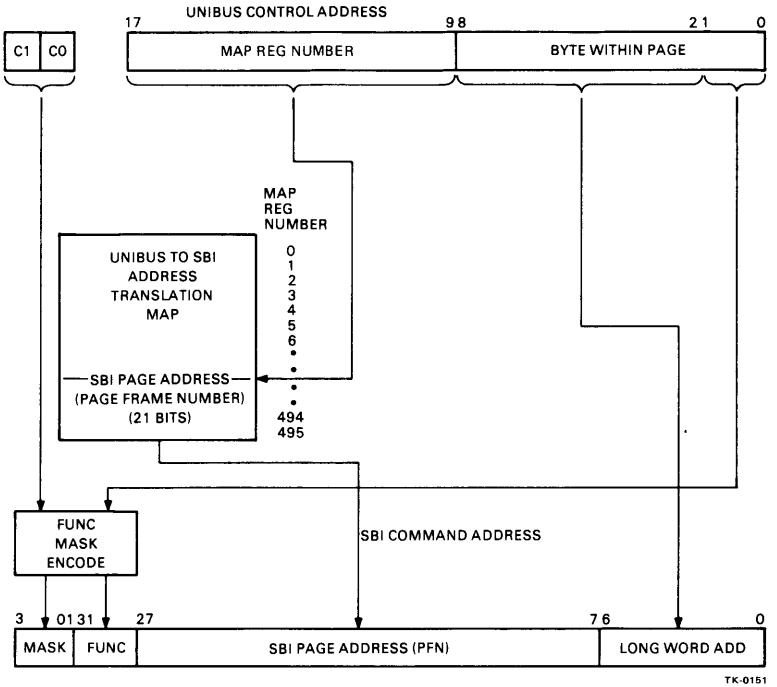
all 2 do stay the same.

SBI TO UNIBUS CONTROL ADDRESS TRANSLATION

SBI COMMAND ADDRESS FORMAT



UNIBUS TO SBI ADDRESS TRANSLATION



ADDRESSES AND VECTORS FOR UNIBUS DEVICES

UBA No.	Device	UNIBUS Address (Octal)	VAX-11/780 Physical Address (Hex)	Vector (Octal)
0	CR11	777160	2013FE70	230
	DR11B/DR11W	772410	2013F508	124
	DMC11/DMR11	Float	--	Float
	DZ11	Float	--	Float
	KMC11	Float	--	Float
	LP11 0	777514	2013FF4C	200
	LP11 1	764004	2013E804	170
	LP11 2	764014	2013E80C	174
	LP11 3	764024	2013E814	270
	LPA11	770460	2013F130	Float
	RK711	777440	2013FF20	210
	RL211	774400	2013F900	160

NOTE: Floating addresses and vectors are according to PDP-11 convention.

FLOATING VECTORS AND FLOATING ADDRESSES

Floating Vectors
(Start at 300 and Proceed Upwards)

Device	Vector Size
DC11	10
DL11-A,-B	10*
DP11	10
DM11-AA	10*
DN11	4
DM11-BB	4
DR11-A	10
DR11-C	10
PA611 Reader	4
PA611 Punch	4
LPD11	
DT11	10*
DX11	10*
DL11C,D,E	10*
DJ11	10*
DH11	10**
GT40	10
LPS11	30*
DQ11	10**
KW11-W	10*
DU11	10*
DUP11	10*
DV11-Data	10*
DV11-Modem Control	4
LK11-A	
DWUN	
DMC-11	
DZ11	
DWR70	
LPP11	
VMV21	
VMV31	
VTV01	
KMC11	
RL11/RLV11***	
RX02	
TS11	
LPA11-K	
IP11/IP300	
DMP11-AD	

* The vector for the device of this type must always be on a (10) octal boundary. (No switch or jumper connection for vector bit 2.)

** The device can have either a M7830 or M7821 interrupt control module. However, it should always be on a (10) octal boundary.

*** Only for second or later device.

Floating Addresses
(Default Address If Nothing Precedes it)

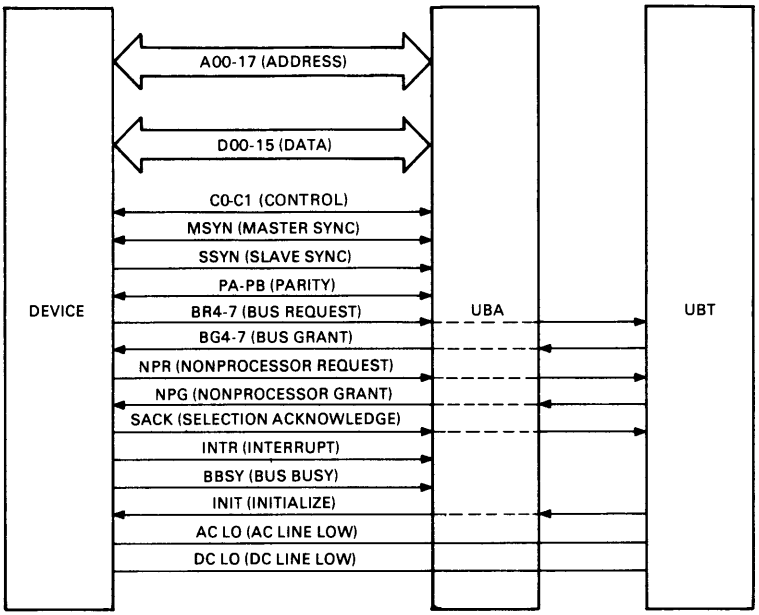
Rank	Device	Address (Octal)	VAX-11/780* Physical Address (Hex)
1	DJ11	760010	2013E008
2	DH11	760020	2013E010
3	DQ11	760030	2013E018
4	DU11	760040	2013E020
5	DUP11	760050	2013E028
6	LK11-A	760060	2013E030
7	DMC11/DMR11	760070	2013E038
8	DZ11	760100	2013E040
9	DWR70	760110	2013E048
10	LPP11	760120	2013E050
11	VMV21	760130	2013E058
12	VMV31	760140	2013E060
13	KMC11	760150	2013E068
14	RL11/RLV11**	760160	2013E070
15	DMP11	760170	2013E078

* This address applies to the UBA at TR3.

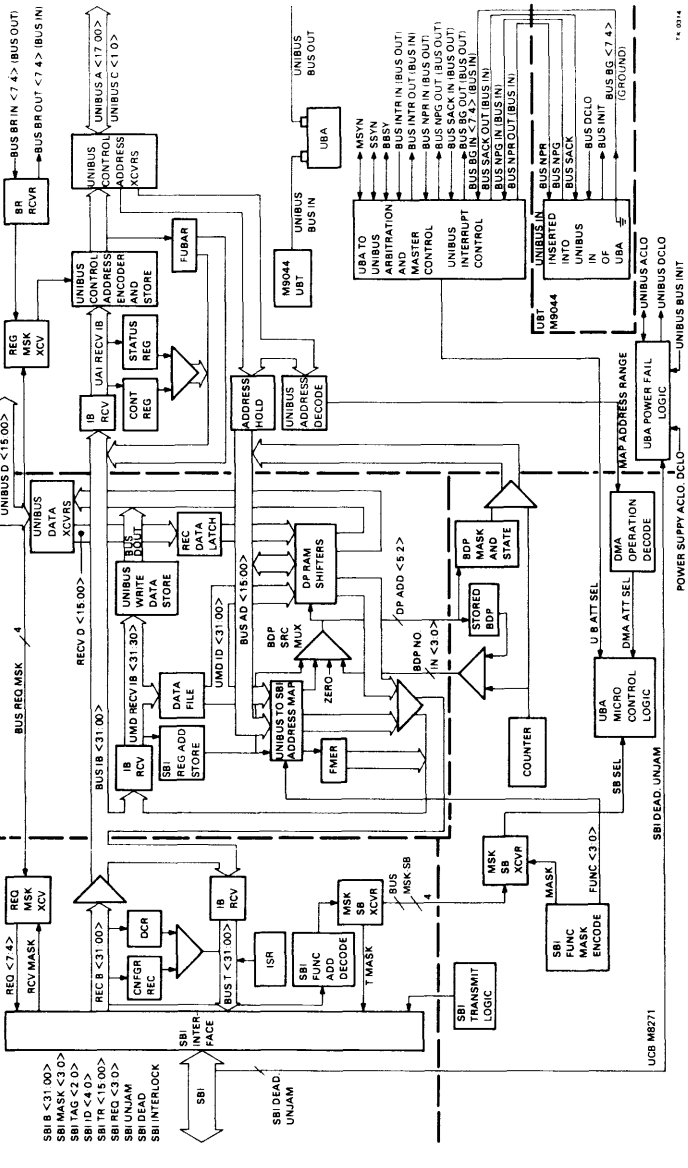
** Only for second or later device.

NOTE: Floating register space begins at address 760010. One 8-byte (10 octal) gap must be left for every device type with floating registers that is not present. In addition, an 8-byte gap must be left after the registers for each device type that is present. Register alignment requirements must be preserved.

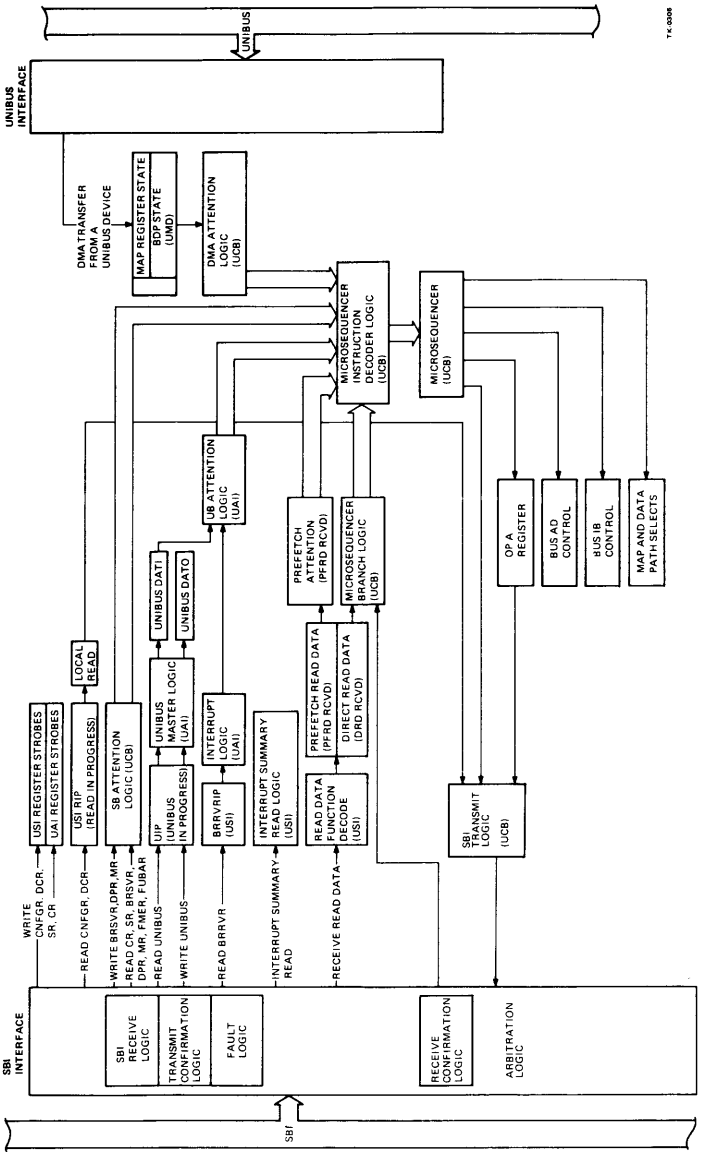
UNIBUS CONFIGURATION



DW780 (UBS) BLOCK DIAGRAM



SIMPLIFIED FLOW OF MAJOR CONTROL FUNCTIONS WITHIN THE UBA



Pin	Standard Signal	Modified Signal	Pin	Standard Signal	Modified Signal	Pin	Standard Signal	Modified Signal
AA1	INIT L	INIT L	AP1	GROUND	P0*	BH1	A01 L	A01 L
AA2	+5 V	+5 V	AP2	BBSY L	BBSY L	BH2	A00 L	A00 L
AB1	INTR L	INTR L	AR1	GROUND	BAT BACKUP +15 V	BJ1	A03 L	A03 L
AB2	GROUND	TEST POINT	AR2	SACK L	SACK L	BJ2	A02 L	A02 L
AC1	D00 L	D00 L	AS1	GROUND	BAT BACKUP +15 V	BK1	A05 L	A05 L
AC2	GROUND	GROUND	AS2	NPR L	NPR L	BK2	A04 L	A04 L
AD1	D02 L	D02 L	AT1	GROUND	GROUND	BL1	A07 L	A07 L
AD2	D01 L	D01 L	AT2	BR7 L	BR7 L	BL2	A06 L	A06 L
AE1	D04 L	D04 L	AU1	NPG H	+20 V	BM1	A09 L	A09 L
AE2	D03 L	D03 L	AU2	BR6 L	BR6 L	BM2	A08 L	A08 L
AF1	D06 L	D06 L	AV1	BG7 H	+20 V	BN1	A11 L	A11 L
AF2	D05 L	D05 L	AV2	GROUND	+20 V	BN2	A10 L	A10 L
AH1	D08 L	D08 L	BA1	BG6 H	SPARE	BP1	A13 L	A13 L
AH2	D07 L	D07 L	BA2	+5 V	+5 V	BP2	A12 L	A12 L
AJ1	D10 L	D10 L	BB1	BG5 H	SPARE	BR1	A15 L	A15 L
AJ2	D09 L	D09 L	BB2	GROUND	TEST POINT	BR2	A14 L	A14 L
AK1	D12 L	D12 L	BC1	BR5 L	BR5 L	BS1	A17 L	A17 L
AK2	D11 L	D11 L	BC2	GROUND	GROUND	BS2	A16 L	A16 L
AL1	D14 L	D14 L	BD1	GROUND	BAT BACKUP +5 V	BT1	GROUND	GROUND
AL2	D13 L	D13 L	BD2	BR4 L	BR4 L	BT2	C1 L	C1 L
AM1	PA L	PA L	BE1	GROUND	INT SSYN*	BU1	SSYN L	SSYN L
AM2	D15 L	D15 L	BE2	BG4 H	PAR DET*	BU2	C0 L	C0 L
AN1	GROUND	P1*	BF1	ACLO L	ACLO L	BV1	MSYN L	MSYN L
AN2	PB L	PB L	BF2	DCLO L	DCLO L	BV2	GROUND	-5 V

*Pins used by parity control module.

UNIBUS SIGNAL DESCRIPTIONS

Signal Line	Description
Data Transfer Group	
Address Lines [SA (17:00)]	These lines are used by the master device to select the slave (actually a unique memory or device register address). SA (17:01) specifies a unique 16-bit word; SA00 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 coded by the master device to control the slave in one of the four possible data transfer operations specified below. Note that the transfer direction is always designated with respect to the master device.
C1 C0	
0 0	Data In (DATI): a data word or byte transferred into 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.
1 1	Data Out Byte (DATOB): 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 asserted. PB, when true, indicates a device parity error.
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 DATOB) 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/P 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 UBA 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 transaction. INTR may be asserted only by a device that obtained bus mastership 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.

UNIBUS SIGNAL DESCRIPTIONS (CONT)

Signal Line	Description
Priority Arbitration Group (Cont)	
Nonprocessor Request (NPR)	This is a bus request from a device for a transfer not requiring CPU intervention (i.e., DMA).
Nonprocessor Grant (NPG)	This is the grant in response to an NPR.
Selection Acknowledge (SACK)	SACK is asserted by a bus-requesting device after having received a grant. Bus control passes to this device when the current bus master completes its operation.
Bus Busy (BBSY)	BBSY indicates that the data lines of the bus are in use. It is asserted by the Unibus master.
Initialization Group	
Initialize (INIT)	This signal is asserted by the terminator board (UBT) when DC LO is asserted on the Unibus, and it stays asserted for 10 ms following the negation of DC LO.
AC Line Low (AC LO)	This is an anticipatory signal that warns of an impending power failure. AC LO initiates the power fail trap sequence and may also be issued in peripheral devices to terminate operations in preparation for power loss.
DC Line Low (DC LO)	This signal is available from each system power supply and remains clear as long as all dc voltages are within the specified limits. If an out-of-voltage condition occurs, DC LO is asserted.

DW780 MODULE UTILIZATION CHART

TYPICAL CONFIGURATION					
6	5	4	3	2	1
BLANK MODULE	BLANK MODULE	UAI M8273	UMD M8272	UCB M8271	USI M8270

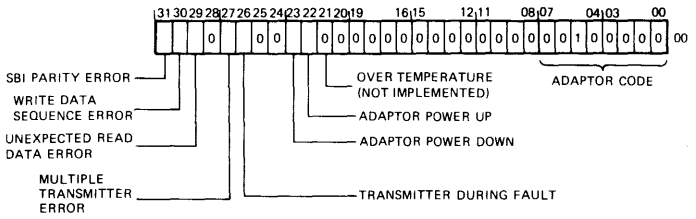
TK-8357

MBA REGISTERS

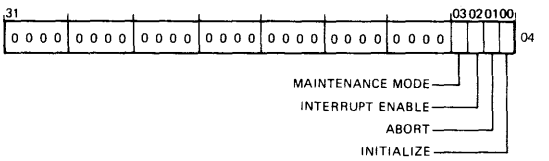
MBA Register Base Address
as a Function of TR Number

TR Num Base 10	Base Address (Physical Hex)	SBI Address (Hex)
1	20002000	8000800
2	20004000	8001000
3	20006000	8001800
4	20008000	8002000
5	2000A000	8002800
6	2000C000	8003000
7	2000E000	8003800
8	20010000	8004000
9	20012000	8004800
10	20014000	8005000
11	20016000	8005800
12	20018000	8006000
13	2001A000	8006800
14	2001C000	8007000
15	2001E000	8007800

MBA CONFIGURATION/STATUS REGISTER



MBA CONTROL REGISTER

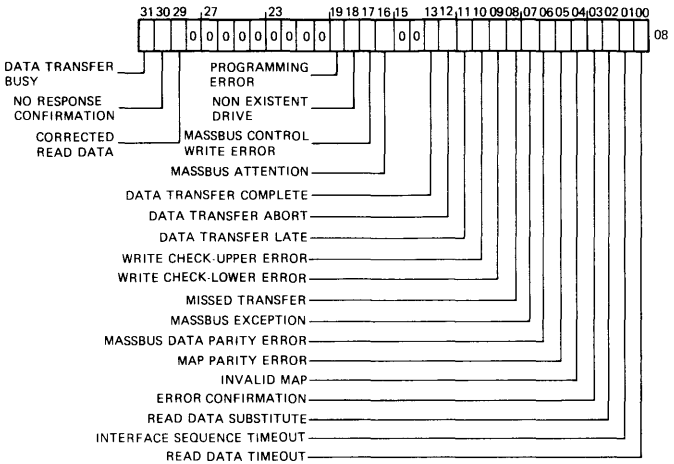


NOTE: ALL BITS ARE READ/WRITE EXCEPT INITIALIZE WHICH ALWAYS READS AS 0

TK-0693

MBA REGISTERS (CONT)

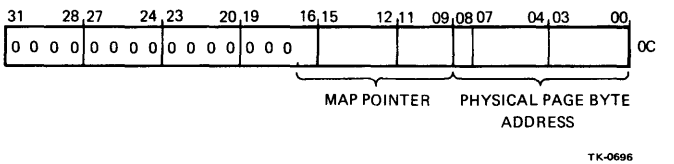
MBA STATUS REGISTER



NOTE: WRITE 1 TO CLEAR BITS IN THIS REGISTER EXCEPT BITS 31 AND 16, WHICH ARE READ ONLY.

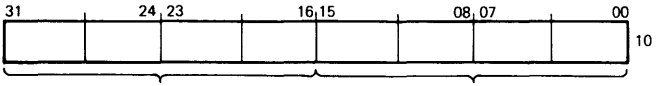
TK-0698

MBA VIRTUAL ADDRESS REGISTER



TK-0696

MBA BYTE COUNT REGISTER



MASSBUS BYTE COUNTER (READ ONLY)

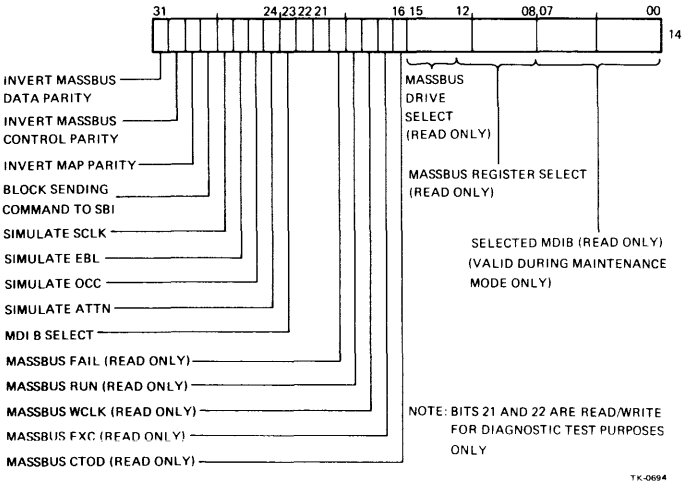
SBI BYTE COUNTER (READ/WRITE)

NOTE: DATA WRITTEN INTO THE SBI BYTE COUNTER IS COPIED INTO THE MASSBUS BYTE COUNTER.

2's COMPLEMENT OF THE NUMBER OF BYTES TO BE TRANSFERRED

TK-0697

MBA DIAGNOSTIC REGISTER

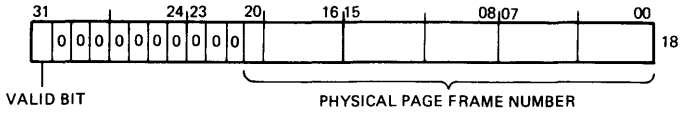


NOTE: BITS 21 AND 22 ARE READ/WRITE FOR DIAGNOSTIC TEST PURPOSES ONLY

TK-0694

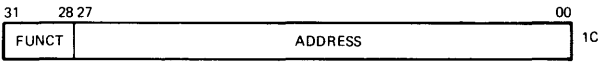
MBA REGISTERS (CONT)

MBA MAP REGISTER



TK-0715

COMMAND/ADDRESS REGISTER (CAR)



*The CAR is read-only, and is only valid when DT BUSY is set. This register contains the value of bits 31 through 00 of the SBI during the command/address portion of the MBA's next data transfer.

TK-8349

b	d	f	j	l	n	r	t	v	x	z	bb	dd	ff	jj	ll	nn	rr	tt	vv	RH780 Configuration
a	c	e	h	k	m	p	s	u	w	y	aa	cc	ee	hh	kk	mm	pp	ss	uu	for REV A Backpanel
W1	W2	W3	W4	W5	W6	W7	W8	W9	W1Ø	W11	W12									
*	*	*	*		*	*		*	*	*	*	*	*							Configuration
*	*	*	*		*	*		*	*	*	*	*	*							for REV -- Backpanel
W1	W2	W3	W4		W5	W6		W7	W8	W9	W1Ø	W11	W12							

TR Arbitration Level

Signal Name	MBA TR SEL D	MBA TR SEL C	MBA TR SEL B	MBA TR SEL A	Wire Wrap Bus SBI TRXX L from FØ2F1 to
TR#	W1	W2	W3	W4	
1	--	--	--	--	FØ2C1
2	--	--	--	I	FØ2D1
3	--	--	I	--	FØ2E1
4	--	--	I	I	FØ2F2
5	--	I	--	--	FØ2H2
6	--	I	--	I	FØ2J1
7	--	I	I	--	FØ2J2
8	--	I	I	I *	FØ2M1
9	I	--	--	--	FØ2N1
1Ø	I	--	--	I	FØ2P1
11	I	--	I	--	FØ2P2
12	I	--	I	I	FØ2S2
13	I	I	--	--	FØ2T2
14	I	I	--	I	FØ2U1
15	I	I	I	--	FØ2U2

Interrupt Level Selection

Signal Name	MBA Intr Code 1 H	MBA INTR Code Ø H
BR#	W5	W6
4	--	--
5	--	I *
6	I	--
7	I	I

W7 - W12 SPARES

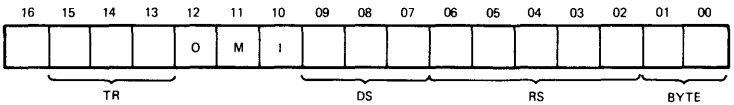
* Normal for first RH78Ø

MASSBUS DISK DRIVE REGISTER ADDRESS CALCULATION CHART

1ST MBA BASE ADDRESS – 20010400 (TR8) 3RD MBA BASE ADDRESS – 20014400 (TR10)
2ND MBA BASE ADDRESS – 20012400 (TR9) 4TH MBA BASE ADDRESS – 20016400 (TR11)

REGISTER NUMBER		DRIVE TYPE			DRIVE NUMBER							
HEX	OCTAL	RP (DISK)	RM (DISK)	TE (TAPE)	0	1	2	3	4	5	6	7
0	0	CSI	RMCS1	CS1	0	80	100	180	200	280	300	380
1	1	DS	RMDS	DS	4	84	104	184	204	284	304	384
2	2	ER1	RMER1	ER	8	88	108	188	208	288	308	388
3	3	MR	RMMR1	MR	C	8C	10C	18C	20C	28C	30C	38C
4	4	AS	RMAS	AS	10	90	110	190	210	290	310	390
5	5	DA	RMDA	FC	14	94	114	194	214	294	314	394
6	6	DT	RMDT	DT	18	98	118	198	218	298	318	398
7	7	LA	RMLA	CX	1C	9C	11C	19C	21C	29C	31C	39C
8	10	SN	RMSN	SN	20	A0	120	1A0	220	2A0	320	3A0
9	11	OFF	RMOF	TC	24	A4	124	1A4	224	2A4	324	3A4
A	12	DCA	RMOC		28	A8	128	1A8	228	2A8	328	3A8
B	13	CCA	RMNR		2C	AC	12C	1AC	22C	2AC	32C	3AC
C	14	ER2	RMMR2		30	B0	130	1B0	230	2B0	330	3B0
D	15	ER3	RMER2		34	B4	134	1B4	234	2B4	334	3B4
E	16	ECCPOS	RMEC1		38	B8	138	1B8	238	2B8	338	3B8
F	17	ECCPAT	RMEC2		3C	BC	13C	1BC	23C	2BC	33C	3BC
•	•				•	•	•	•	•	•	•	•
•					•	•	•	•	•	•	•	•
					•	•	•	•	•	•	•	•
1F	37				7C	FC	17C	1FC	27C	2FC	37C	3FC

MBA BASE PHYSICAL ADDRESS TRANSLATION



M = MAP REG SELECT
I = SET IF EXTERNAL REGISTER
DS = DRIVE SELECT #
RS = REGISTER SELECT
0 = ZERO

TK-8347

MASSBUS SIGNAL CABLE PIN ASSIGNMENTS

Massbus Signal Cable Designations

Cable	Pin*		Polarity	Designation
Massbus Cable A	A	1	-	MASS D00
	B	2	+	
	C	3	+	MASS D01
	D	4	-	
	E	5	-	MASS D02
	F	6	+	
	H	7	+	MASS D03
	J	8	-	
	K	9	-	MASS D04
	L	10	+	
	M	11	+	MASS D05
	N	12	-	
	P	13	-	MASS C00
	R	14	+	
	S	15	+	MASS C01
	T	16	-	
	U	17	-	MASS C02
	V	18	+	
	W	19	+	MASS C03
	X	20	-	
	Y	21	-	MASS C04
	Z	22	+	
	AA	23	+	MASS C05
	BB	24	-	
	CC	25	-	MASS SCLK
	DD	26	+	
	EE	27	+	MASS RS3
	FF	28	-	
	HH	29	+	MASS ATTN
	JJ	30	-	
	KK	31	-	MASS RS4
	LL	32	+	
	MM	33	-	MASS CTOD
	NN	34	+	
	PP	35	+	MASS WCLK
	RR	36	-	
	SS	37	+	MASS RUN
	TT	38	-	
	UU	39		SPARE
	VV	40		GND

Massbus Signal Cable Designations

Cable	Pin*		Polarity	Designation
Massbus Cable B	A	1	-	MASS D06
	B	2	+	
	C	3	+	MASS D07
	D	4	-	
	E	5	-	MASS D08
	F	6	+	
	H	7	+	MASS D09
	J	8	-	
	K	9	-	MASS D10
	L	10	+	
	M	11	+	MASS D11
	N	12	-	
	P	13	-	MASS C06
	R	14	+	
	S	15	+	MASS C07
	T	16	-	
	U	17	-	MASS C08
	V	18	+	
	W	19	+	MASS C09
	X	20	-	
	Y	21	-	MASS C10
	Z	22	+	
	AA	23	+	MASS C11
	BB	24	-	
	CC	25	-	MASS EXC
	DD	26	+	
	EE	27	+	MASS RS0
	FF	28	-	
	HH	29	+	MASS EBL
	JJ	30	-	
	KK	31	-	MASS RS1
	LL	32	+	
	MM	33	-	MASS RS2
	NN	34	+	
	PP	35	+	MASS INIT
	RR	36	-	
	SS	37	+	MASS SP1
	TT	38	-	
	UU	39		SPARE
	VV	40		GND

*Alternate pin designation schemes

Note: Massbus cables are to be installed per markings on the cable.

MASSBUS SIGNAL CABLE PIN ASSIGNMENTS (CONT)

Massbus Signal Cable Designations

Cable	Pin*		Polarity	Designation
Massbus Cable C	A	1	-	MASS D12
	B	2	+	
	C	3	+	MASS D13
	D	4	-	
	E	5	-	MASS D14
	F	6	+	
	H	7	+	MASS D15
	J	8	-	
	K	9	-	MASS D16
	L	10	+	
	M	11	+	MASS D17
	N	12	-	
	P	13	-	MASS DPA
	R	14	+	
	S	15	+	MASS C12
	T	16	-	
	U	17	-	MASS C13
	V	18	+	
	W	19	+	MASS C14
	X	20	-	
	Y	21	-	MASS C15
	Z	22	+	
	AA	23	+	MASS CPA
	BB	24	-	
	CC	25	-	MASS OCC
	DD	26	+	
	EE	27	+	MASS DS0
	FF	28	-	
	HH	29	+	MASS TRA
	JJ	30	-	
	KK	31	-	MASS DS1
	LL	32	+	
	MM	33	-	MASS DS2
	NN	34	+	
	PP	35	+	MASS DEM
	RR	36	-	
	SS	37	+	MASS SP2
	TT	38	-	
	UU	39	H	MASS FAIL
	VV	40		GND

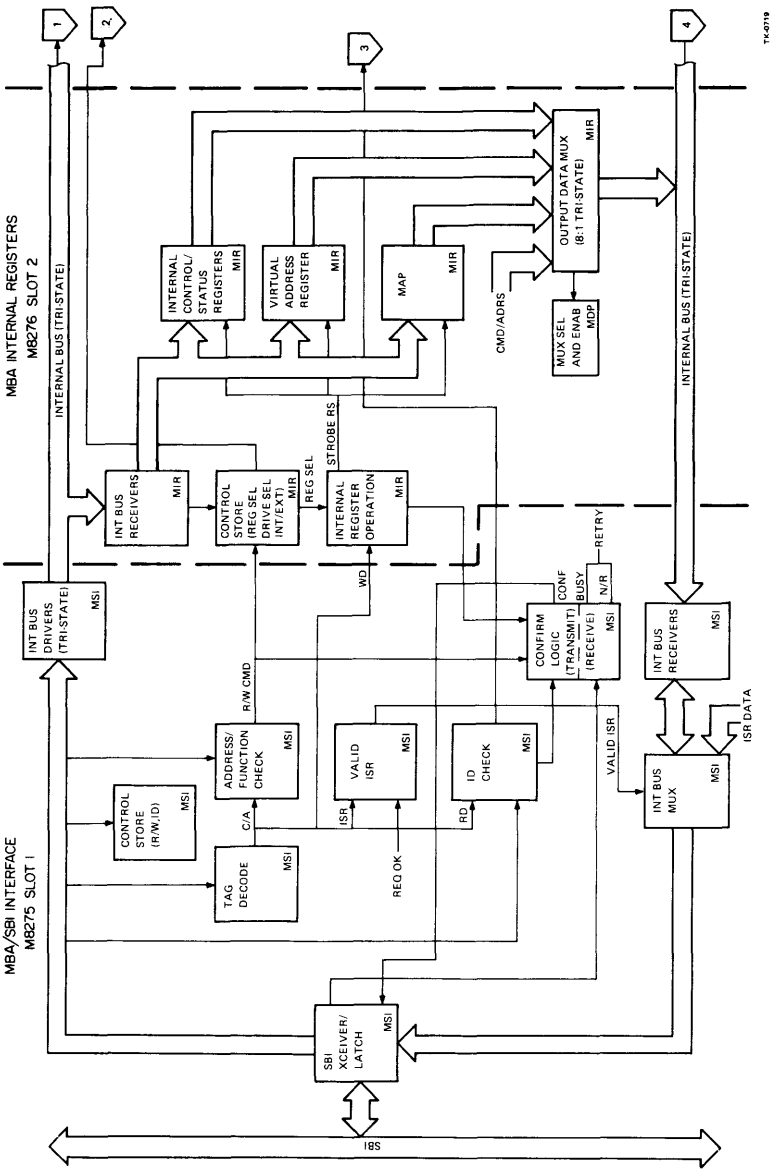
*Alternate pin designation schemes

RH780 MODULE UTILIZATION CHART

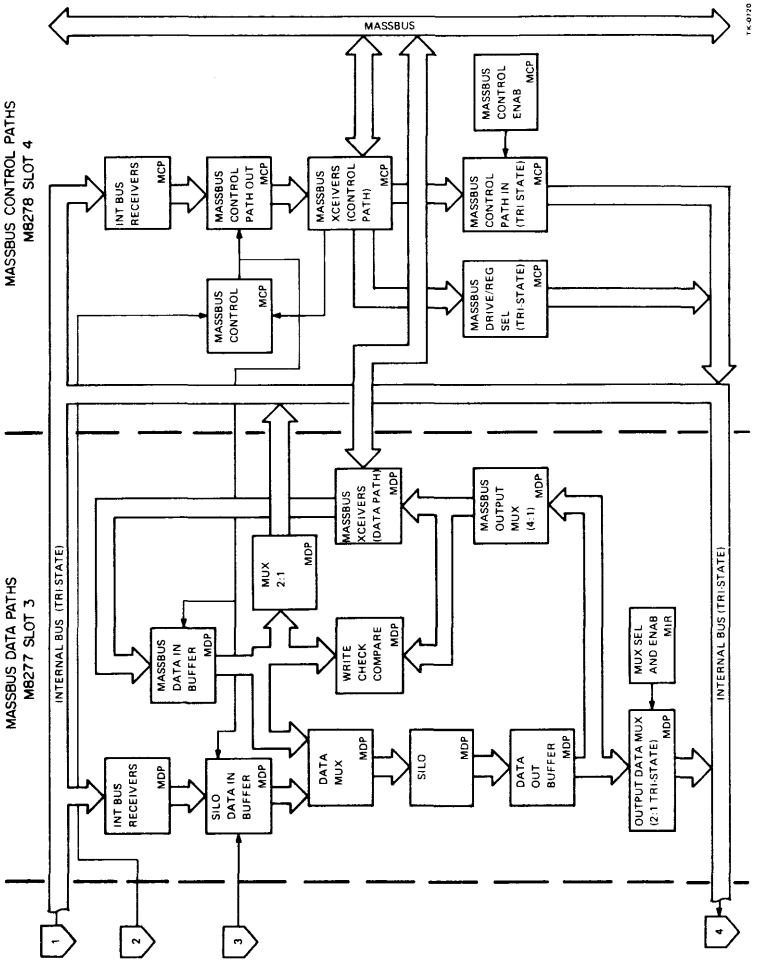
TYPICAL CONFIGURATION					
6	5	4	3	2	1
BLANK MODULE	BLANK MODULE	MCP M8278	MDP M8277	MIR M8276	MSI M8275

TK-8356

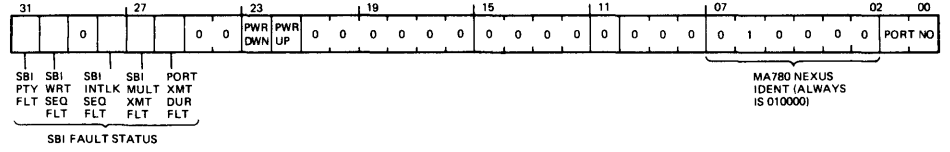
MBA (RH780) BLOCK DIAGRAM, PART 1



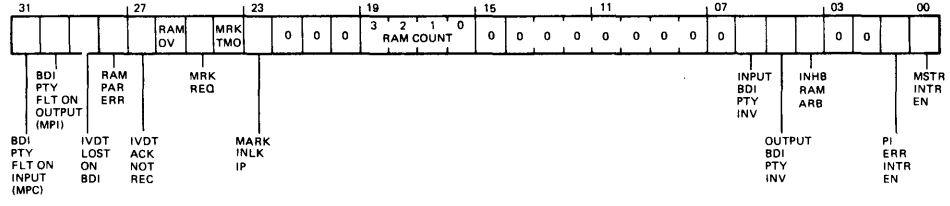
MBA (RH780) BLOCK DIAGRAM, PART 2



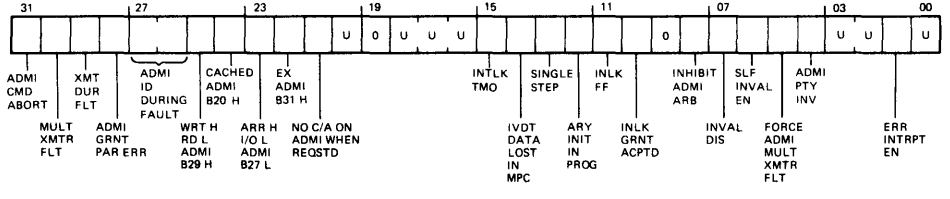
PORT CONFIGURATION REGISTER 2000X000 MPI



PORT INTERFACE CONTROL REGISTER 2000X004 MPI



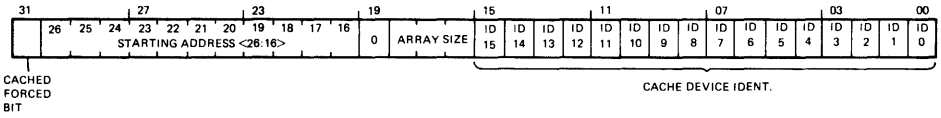
PORT CONTROLLER STATUS REGISTER 2000X008 MPC



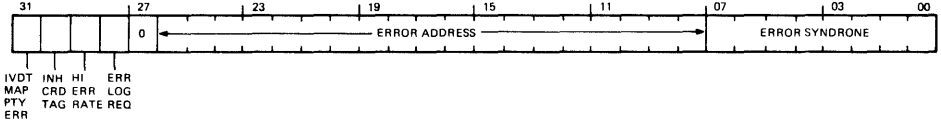
TR. NO.	X=
1	2
2	4
3	6
4	8

TK-3394

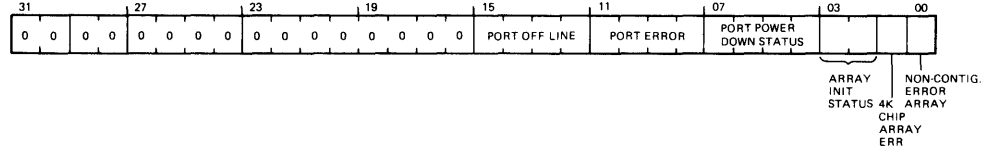
PORT INVALIDATION CONTROL REGISTER 2000X00C MPC



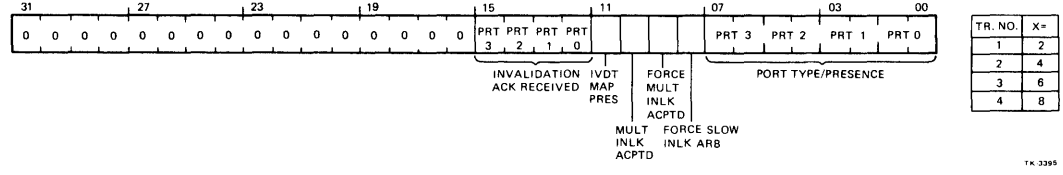
ARRAY ERROR REGISTER 2000X010 MAT



CONFIGURATION STATUS REGISTER 0 2000X014 MAT



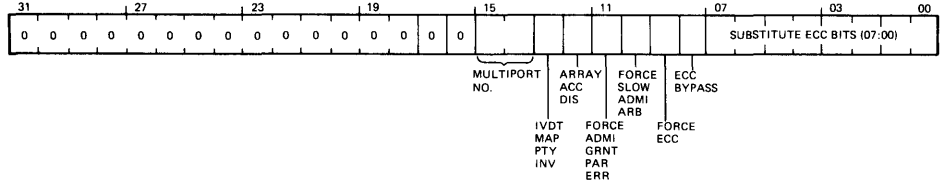
CONFIGURATION STATUS REGISTER 1 2000X018 MPS



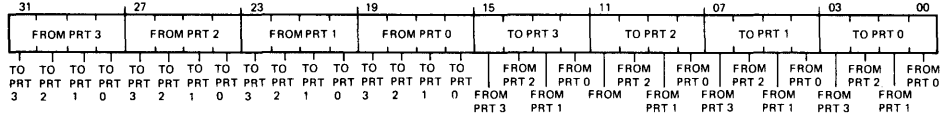
TR. NO.	X=
1	2
2	4
3	6
4	8

TK 3395

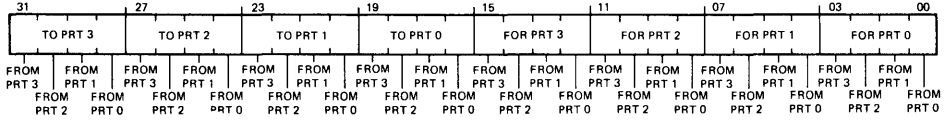
MAINTENANCE CONTROL REGISTER 2000X01C MAT



INTERPORT INTERRUPT REQUEST REGISTER 2000X020 MPS



INTERPORT INTERRUPT ENABLE REGISTER 2000X024 MPS



TR NO.	X=
1	2
2	4
3	6
4	8

TK 3420

MA780 ARRAY ADDRESSES

Array	M8210	Address Range
1	256K	0 - 3FFFF
2	512K	40000 - 7FFFF
3	768K	80000 - BFFFF
4	1024K	C0000 - FFFFF
5	1280K	100000 - 13FFFF
6	1536K	140000 - 17FFFF
7	1792K	180000 - 1BFFFF
8	2048K	1C0000 - 1FFFFF

MA780C JUMPERS

W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
B	D	F	J	L	N	R	T	U	X	Z	B	D	F	J	L	N	R	T	V
											B	D	F	J	L	N	R	T	V

SBI TR Level Jumpers

Standard Port Interface Slots 3 and 4

TR Level	Jumper				Wire Wrap
	W17	W18	W19	W20	F03H1 to
1	-	-	-	-	F03C1
2	-	-	-	I	F03D1 *
3	-	-	I	-	F03E1
4	-	-	I	I	F03F2
5	-	I	-	-	F03H2
6	-	I	-	I	F03J1
7	-	I	I	-	F03J2
8	-	I	I	I	F03M1
9	I	-	-	-	F03N1
10	I	-	-	I	F03P1
11	I	-	I	-	F03P2
12	I	-	I	I	F03S2
13	I	I	-	-	F03T2
14	I	I	-	I	F03U1
15	I	I	I	-	F03U2

Optional Port Interface Slots 1 and 2

Jumper				Wire Wrap
W4	W3	W2	W1	F02H1 to
-	-	-	-	F02C1
-	-	-	I	F02D1
-	-	I	-	F02E1 *
-	-	I	I	F02F2
-	I	-	-	F02H2
-	I	-	I	F02J1
-	I	I	-	F02J2
-	I	I	I	F02M1
I	-	-	-	F02N1
I	-	-	I	F02P1
I	-	I	-	F02P2
I	-	I	I	F02S2
I	I	-	-	F02T2
I	I	-	I	F02U1
I	I	I	-	F02U2

The memory that contains the ROM bootstrap must be at TR 1.

- MA780(s) must have the next highest SBI priority, that is, lowest TR number.
- MS780(s) have the next highest.
- DW780(s) have the next highest.
- RH780(s) have the next highest.

If a MS780 contains the ROM bootstrap, it must be at TR 1. If MS780 memories are on the system and are to be interleaved, the second MS780 must be at the next even TR number past the last MA780 TR.

Interrupt Level Jumpers

Interport Interrupt Level	Error Interrupt Level	Standard Port Slots 3 and 4	Optional Port Slots 1 and 2
		W16	W5
4	5	- *	-
6	7	I	I

I = Jumper inserted.
- = No jumper.
* = Standard configuration.

MA780A JUMPERS

W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
B	D	F	J	L	N	R	T	U	X	Z	B	D	F	J	L	N	R	T	V
											B	D	F	J	L	N	R	T	V

Memory Starting Address Jumpers
(Used Only on Power-Up)

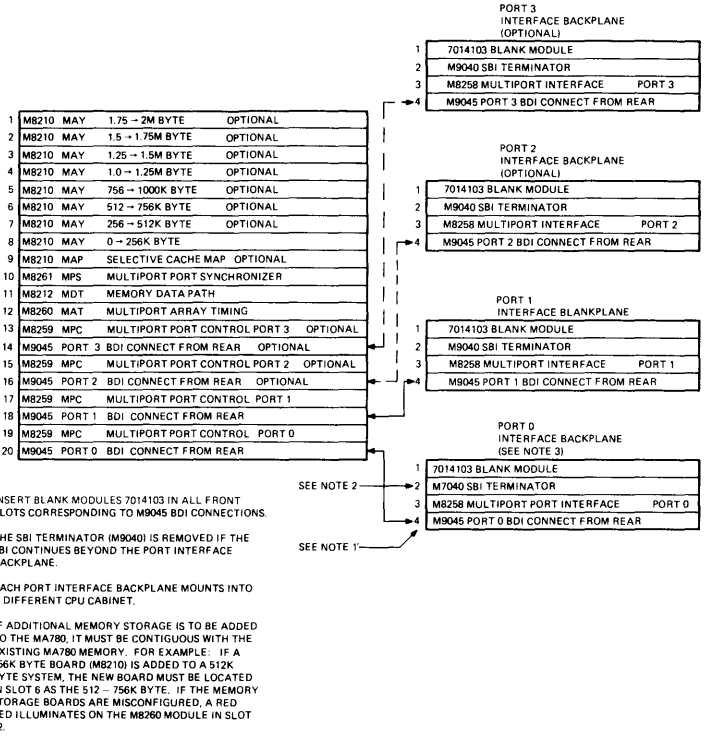
Strap Name	Port 0 SA23	Port 0 SA22	Port 1 SA23	Port 1 SA22	Port 2 SA23	Port 2 SA22	Port 3 SA23	Port 3 SA22	
Jumper	W19	W20	W15	W16	W11	W12	W7	W8	
0MB	I	I	I	I	I	I	I	I	
16MB	-	-	-	-	-	-	-	-	*
20MB	-	I	-	I	-	I	-	I	
24MB	I	-	I	-	I	-	I	-	

Multiport Number Jumpers
(Number Visible on Control Panel)

Signal	Set Multiport 1	Set Multiport 0	
Jumper	W1	W2	
0	I	I	*
1	-	I	
2	I	-	
3	-	-	

I = Jumper inserted.
- = No jumper.
* = Standard configuration.

MA780 BACKPLANE DATA



NOTES:

1. INSERT BLANK MODULES 7014103 IN ALL FRONT SLOTS CORRESPONDING TO M9045 BDI CONNECTIONS.
2. THE SBI TERMINATOR (M9040) IS REMOVED IF THE SBI CONTINUES BEYOND THE PORT INTERFACE BACKPLANE.
3. EACH PORT INTERFACE BACKPLANE MOUNTS INTO A DIFFERENT CPU CABINET.
4. IF ADDITIONAL MEMORY STORAGE IS TO BE ADDED TO THE MA780, IT MUST BE CONTIGUOUS WITH THE EXISTING MA780 MEMORY. FOR EXAMPLE: IF A 256K BYTE BOARD (M8210) IS ADDED TO A 512K BYTE SYSTEM, THE NEW BOARD MUST BE LOCATED IN SLOT 6 AS THE 512 – 756K BYTE. IF THE MEMORY STORAGE BOARDS ARE MISCONFIGURED, A RED LED ILLUMINATES ON THE M8260 MODULE IN SLOT 12.

TK-8358

M8210 MEMORY ARRAY CARD MNEMONICS

The following table shows the cross-correlation between the mnemonic when the card is used in the MS780 main memory and the MA780 multiport memory subsystem of the VAX-11/780 system. The M8210 array card, in addition to its identical memory storage function in both memories, is used (one card) as the invalidate map of the multiport memory.

As Array Card in MS780 Main Memory	As Array Card in MA780 Multiport Memory Pin	As Invalidate Map in MA780 Multiport Memory Pin
BUS ENAB ARRAY OUT H	EL1 MATJ ARY OUT EN H	EL1 MPSK INVAL OUT EN H
BUS MOS DAT C00 H		AJ1 BUS INVAL DAT P00 H
BUS MOS DAT C01 H		AH2 BUS INVAL DAT P01 H
BUS MOS DAT C02 H		AF1 BUS INVAL DAT P02 H
BUS MOS DAT C03 H		AE1 BUS INVAL DAT P03 H
BUS MOS DAT C04 H		DL1 BUS INVAL DAT P04 H
BUS MOS DAT C05 H		EB1 BUS INVAL DAT P05 H
BUS MOS DAT C06 H		DP2 BUS INVAL DAT P06 H
BUS MOS DAT C07 H		DR2 BUS INVAL DAT P07 H
BUS MOS DAT L00 H		AN1 BUS INVAL DAT B00 L
BUS MOS DAT L01 H		AM1 BUS INVAL DAT B01 L
BUS MOS DAT L02 H		AL1 BUS INVAL DAT B02 L
BUS MOS DAT L03 H		AK1 BUS INVAL DAT B03 L
BUS MOS DAT L04 H		BA1 BUS INVAL DAT B04 L
BUS MOS DAT L05 H		AV2 BUS INVAL DAT B05 L
BUS MOS DAT L06 H		AU1 BUS INVAL DAT B06 L
BUS MOS DAT L07 H		AR1 BUS INVAL DAT B07 L
BUS MOS DAT L08 H		BE1 BUS INVAL DAT B08 L
BUS MOS DAT L09 H		BD1 BUS INVAL DAT B09 L
BUS MOS DAT L10 H		BC1 BUS INVAL DAT B10 L
BUS MOS DAT L11 H		BB1 BUS INVAL DAT B11 L
BUS MOS DAT L12 H		BK1 BUS INVAL DAT B12 L
BUS MOS DAT L13 H		BJ1 BUS INVAL DAT B13 L
BUS MOS DAT L14 H		BH2 BUS INVAL DAT B14 L
BUS MOS DAT L15 H		BF1 BUS INVAL DAT B15 L
BUS MOS DAT L16 H		BP1 BUS INVAL DAT B16 L
BUS MOS DAT L17 H		BN1 BUS INVAL DAT B17 L
BUS MOS DAT L18 H		BM1 BUS INVAL DAT B18 L
BUS MOS DAT L19 H		BL1 BUS INVAL DAT B19 L
BUS MOS DAT L20 H		BV2 BUS INVAL DAT B20 L
BUS MOS DAT L21 H		BU2 BUS INVAL DAT B21 L
BUS MOS DAT L22 H		BS1 BUS INVAL DAT B22 L
BUS MOS DAT L23 H		DR1 BUS INVAL DAT B23 L
BUS MOS DAT L24 H		CK2 BUS INVAL DAT B24 L
BUS MOS DAT L25 H		CJ2 BUS INVAL DAT B25 L
BUS MOS DAT L26 H		CM1 BUS INVAL DAT B26 L
BUS MOS DAT L27 H		CK1 BUS INVAL DAT B27 L
BUS MOS DAT L28 H		BH2 BUS INVAL DAT B28 L
BUS MOS DAT L29 H		CF2 BUS INVAL DAT B29 L
BUS MOS DAT L30 H		CE2 BUS INVAL DAT B30 L
BUS MOS DAT L31 H		CD2 BUS INVAL DAT B31 L
BUS MOS DAT U00 H		DV2 BUS INVAL DAT B32 L
BUS MOS DAT U01 H		DU2 BUS INVAL DAT B33 L
BUS MOS DAT U02 H		DT2 BUS INVAL DAT B34 L
BUS MOS DAT U03 H		DS2 BUS INVAL DAT B35 L
BUS MOS DAT U04 H		ED1 BUS INVAL DAT B36 L
BUS MOS DAT U05 H		EE1 BUS INVAL DAT B37 L
BUS MOS DAT U06 H		EF2 BUS INVAL DAT B38 L
BUS MOS DAT U07 H		EH2 BUS INVAL DAT B39 L
BUS MOS DAT U08 H		EJ2 BUS INVAL DAT B40 L
BUS MOS DAT U09 H		EK2 BUS INVAL DAT B41 L
BUS MOS DAT U10 H		EL2 BUS INVAL DAT B42 L
BUS MOS DAT U11 H		EM2 BUS INVAL DAT B43 L

M8210 MEMORY ARRAY CARD MNEMONICS (CONT)

As Array Card in MS780 Main Memory		As Array Card in MA780 Multiport Memory Pin	As Invalidate Map in MA780 Multiport Memory Pin
BUS MOS DAT U12 H			ER2 BUS INVAL DAT B44 L
BUS MOS DAT U13 H			ES2 BUS INVAL DAT B45 L
BUS MOS DAT U14 H			EP2 BUS INVAL DAT B46 L
BUS MOS DAT U15 H			EF1 BUS INVAL DAT B47 L
BUS MOS DAT U16 H			EV2 BUS INVAL DAT B48 L
BUS MOS DAT U17 H			EU1 BUS INVAL DAT B49 L
BUS MOS DAT U18 H			EU2 BUS INVAL DAT B50 L
BUS MOS DAT U19 H			ET2 BUS INVAL DAT B51 L
BUS MOS DAT U20 H			FJ2 BUS INVAL DAT B52 L
BUS MOS DAT U21 H			FH2 BUS INVAL DAT B53 L
BUS MOS DAT U22 H			FF2 BUS INVAL DAT B54 L
BUS MOS DAT U23 H			FE2 BUS INVAL DAT B55 L
BUS MOS DAT U24 H			FP2 BUS INVAL DAT B56 L
BUS MOS DAT U25 H			FP1 BUS INVAL DAT B57 L
BUS MOS DAT U26 H			FM2 BUS INVAL DAT B58 L
BUS MOS DAT U27 H			FL2 BUS INVAL DAT B59 L
BUS MOS DAT U28 H			FU2 BUS INVAL DAT B60 L
BUS MOS DAT U29 H			FV2 BUS INVAL DAT B61 L
BUS MOS DAT U30 H			FT2 BUS INVAL DAT B62 L
BUS MOS DAT U31 H			FS2 BUS INVAL DAT B63 L
BUS OUT SEL L	CD1	GND	CD1 GND
MAY CHIP 16K L	FR1	BUS 16K CHIP L	FR1 BUS 16K CHIP L
MAY CHIP 4K L	FS1	BUS 4K CHIP L	FS1 BUS 4K CHIP L
MAY IN 15 L	FB2	BUS PRES BIT 07 H	FB2 INVAL MAP PRES L
MCNB CAS L	CJ1	MATJ ARY CAS L	CJ1 MPSK INVAL CAS L
MCNB INIT H	DB1	MATR INIT H	DB1 MATR INIT H
MCNB MUX CNTRL H	CL1	MATJ ARY MUX CNTRL H	CL1 MBSK INVAL MUX CNTRL L
MCNB READ H	DL2	MATJ ARY READ H	DL2 MPSK INVAL READ H
MCND ARY EXT H	DA1	MATA ARY ADR 14 H	DA1 MATA ARY ADR 14 H
MCND ARY ADR 01 H	DB2	MATA ARY ADR 01 H	DB2 MATA ARY ADR 01 H
MCND ARY ADR 02 H	CV2	MATA ARY ADR 02 H	CV2 MATA ARY ADR 02 H
MCND ARY ADR 03 H	DJ2	MATA ARY ADR 03 H	DJ2 MATA ARY ADR 03 H
MCND ARY ADR 04 H	DH2	MATA ARY ADR 04 H	DH2 MATA ARY ADR 04 H
MCND ARY ADR 05 H	CT2	MATA ARY ADR 05 H	CT2 MATA ARY ADR 05 H
MCND ARY ADR 06 H	DK2	MATA ARY ADR 06 H	DK2 MATA ARY ADR 06 H
MCND ARY ADR 07 H	DF1	MATA ARY ADR 08 H	DF1 MATA ARY ADR 08 H
MCND ARY ADR 08 H	DF2	MATA ARY ADR 09 H	DF2 MATA ARY ADR 09 H
MCND ARY ADR 09 H	CR2	MATA ARY ADR 10 H	CR2 MATA ARY ADR 10 H
MCND ARY ADR 10 H	CS2	MATA ARY ADR 11 H	CS2 MATA ARY ADR 11 H
MCND ARY ADR 11 H	DD1	MATA ARY ADR 12 H	DD1 MATA ARY ADR 12 H
MCND ARY ADR 12 H	DE1	MATA ARY ADR 13 H	DE1 MATA ARY ADR 13 H
MCND ARY ADR 13 H	DM2	MATA ARY ADR 15 H	DM2 MATA ARY ADR 15 H
MCND ARY ADR 16 H	CM2	MATA ARY ADR 16 H	CM2 GND
MCND ARY ADR 17 H	CP2	MATA ARY ADR 17 H	CP2 GND
MCND ARY ADR 18 H	CL2	MATA ARY ADR 18 H	CL2 GND
MCND ARY ADR 19 H			CN1 GND
MCND ARY CS L	DC1	MATA ARY ADR 07 H	DC1 MATA ARY ADR 07 H
MCND RAS L	CP1	MATJ ARY RAS L	CP1 MPSK INVAL RAS L
MCND REF CYC H	DJ1	MATK REF CYC H	DJ1 MPSK INVAL REF CYC H

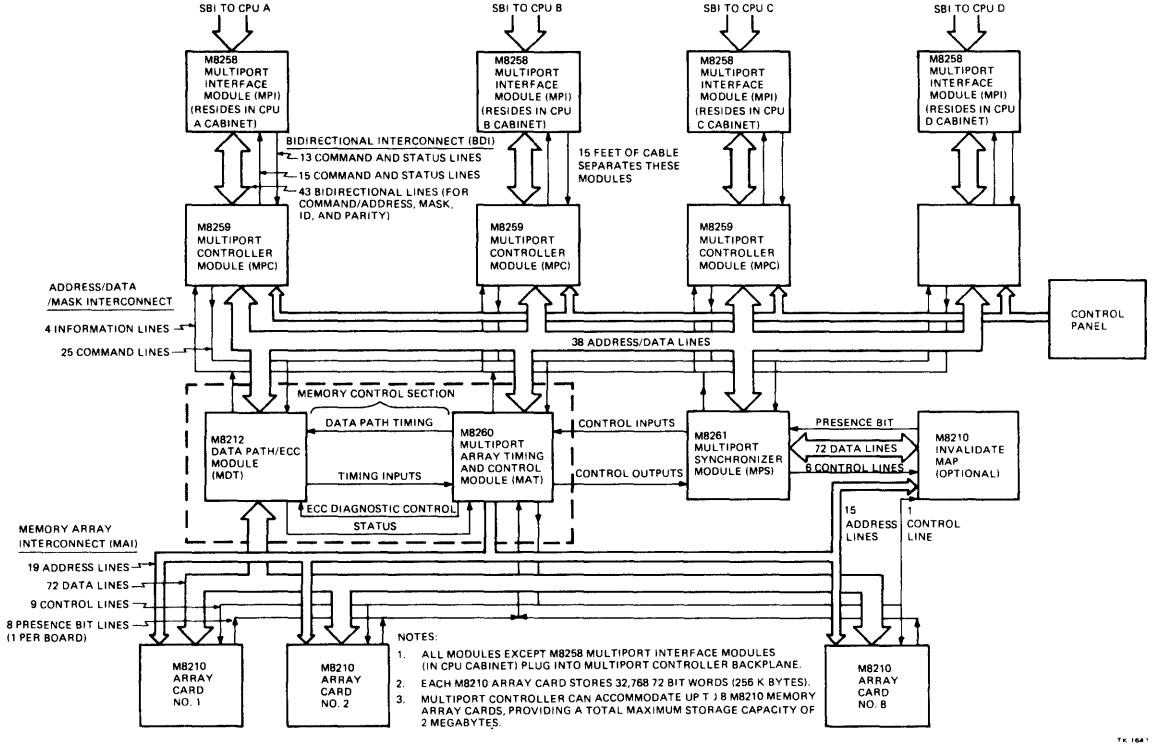
M8212 DATA PATH/ECC CARD MNEMONICS

The following table of signal mnemonics for the M8212 data path/ECC shows the cross-correlation between the mnemonic when the card is used in the MS78Ø main memory and the MA78Ø multiport memory subsystem of the VAX-11/78Ø system. The M8212 serves an identical function in both memories.

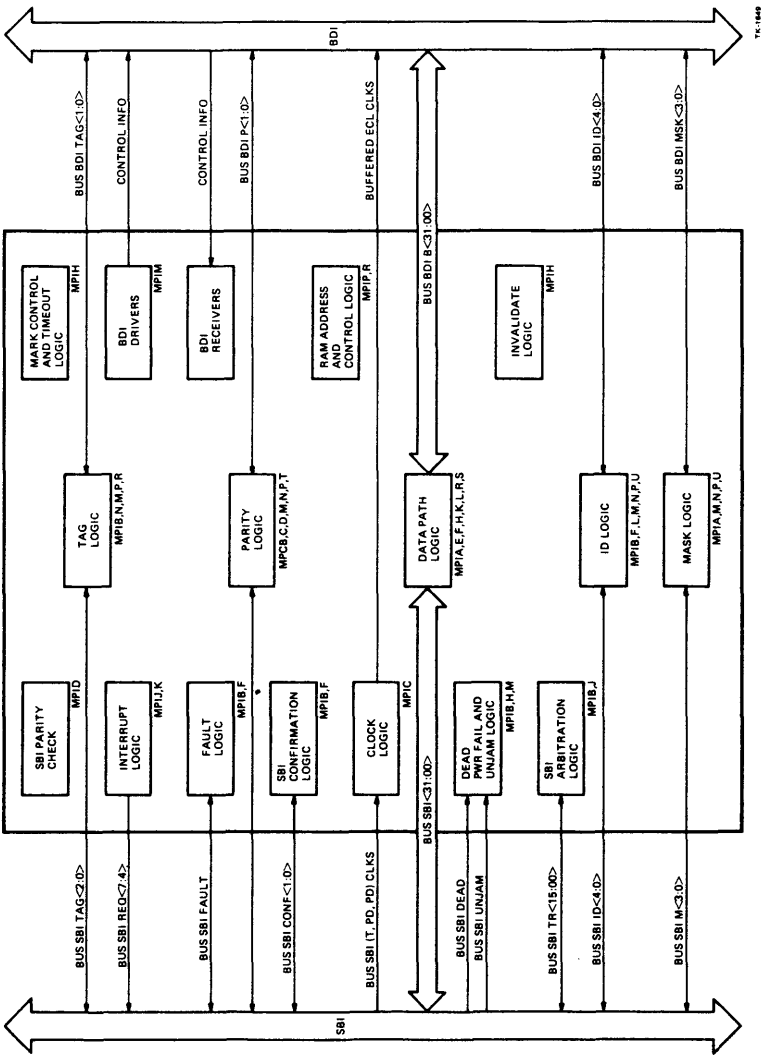
As Used in MS78Ø Main Memory	As Used in MA78Ø Multiport Memory
MCNA ARY RD EN H	DU1 MATK ARY RD EN H
MCNA CLR L	EB1 MATJ CYC CLR L
MCNA ECC EN G	FU1 MATJ ECC EN H
MCNA RD DA1 EN L	ED2 MATK RD DAT EN L
MCNA RD LO SEL L	EC1 MATK RD LO SEL L
MCNA WR EN LO H	EJ1 MATJ WR EN LO H
MCNA WR EN UP H	EF1 MATJ WR EN UP H
MCNB ARY DAT CK H	DM1 MATJ ARY DAT CK H
MCNB CHK BIT DAT CK H	DD2 MATJ CHK BIT DAT CK H
MCNB CHK DAT CK H	DD2 MATJ CHK BIT DAT CK H
MCNB INIT H	DB2 MATR INIT H
MCNB MSK CLR L	DF2 MATJ MSK CLR L
MCNB RD EN H	EL1 MATJ RD EN H
MCNB READ H	DJ1 MATJ GEN ECC L
MCNB TAG CLK L	DF1 MATJ TAG CLK L
MCNB UNCOR ERR CLK L	DE2 MATJ UNCORR ERR CLK L
MCNB XMT DAT CK H	EA1 MATJ XMT DAT CK H
MCNE CRD INH L	EM1 MATF CRD INH L
MCNE DIAG EN L	FS1 MATE DIAG EN L
MCNL FRC CHK O H	FR1 MATD FRC CHK O H
MCNL FRC CHK 1 H	FP1 MATD FRC CHK 1 H
MCNL FRC CHK 2 H	FN1 MATD FRC CHK 2 H
MCNL FRC CHK 3 H	FM1 MATD FRC CHK 3 H
MCNL FRC CHK 4 H	FL1 MATD FRC CHK 4 H
MCNL FRC CHK 5 H	FK2 MATD FRC CHK 5 H
MCNL FRC CHK 6 H	FK1 MATD FRC CHK 6 H
MCNL FRC CHK 7 H	FJ1 MATR FRC CHK 7 H
MDTC SYN 2 H	FD1 MDTA SYN 2 H
MDTE FULL WR EN H	DJ2
MSBH ADR O BUS L	CM1 MATH ADR ON BUS L
MSBH DAT O BUS L	CL1 MATH DAT ON BUS L
MSBJ FL EXT H	DR1 BUS ADMI EXT H
MSBJ FL WR H	DP1 BUS ADMI B29 H
BUS CMD ARY H	DS1 SUB ADMI B29 H
BUS FL INF ØØ H	AA1 BUS ADMI BØØ H
BUS FL INF Ø1 H	AB2 BUS ADMI BØ1 H
BUS FL INF Ø2 H	AD2 BUS ADMI BØ2 H
BUS FL INF Ø3 H	AE2 BUS ADMI BØ3 H
BUS FL INF Ø4 H	AF2 BUS ADMI BØ4 H
BUS FL INF Ø5 H	AJ2 BUS ADMI BØ5 H
BUS FL INF Ø6 H	AK2 BUS ADMI BØ6 H
BUS FL INF Ø7 H	AL2 BUS ADMI BØ7 H
BUS FL INF Ø8 H	AM2 BUS ADMI BØ8 H
BUS FL INF Ø9 H	AR2 BUS ADMI BØ9 H
BUS FL INF 1Ø H	AV2 BUS ADMI B1Ø H
BUS FL INF 11 H	BA1 BUS ADMI B11 H
BUS FL INF 12 H	BB2 BUS ADMI B12 H
BUS FL INF 13 H	BD2 BUS ADMI B13 H
BUS FL INF 14 H	BE2 BUS ADMI B14 H
BUS FL INF 15 H	BK2 BUS ADMI B15 H

M8212 DATA PATH/ECC CARD MNEMONICS (CONT)

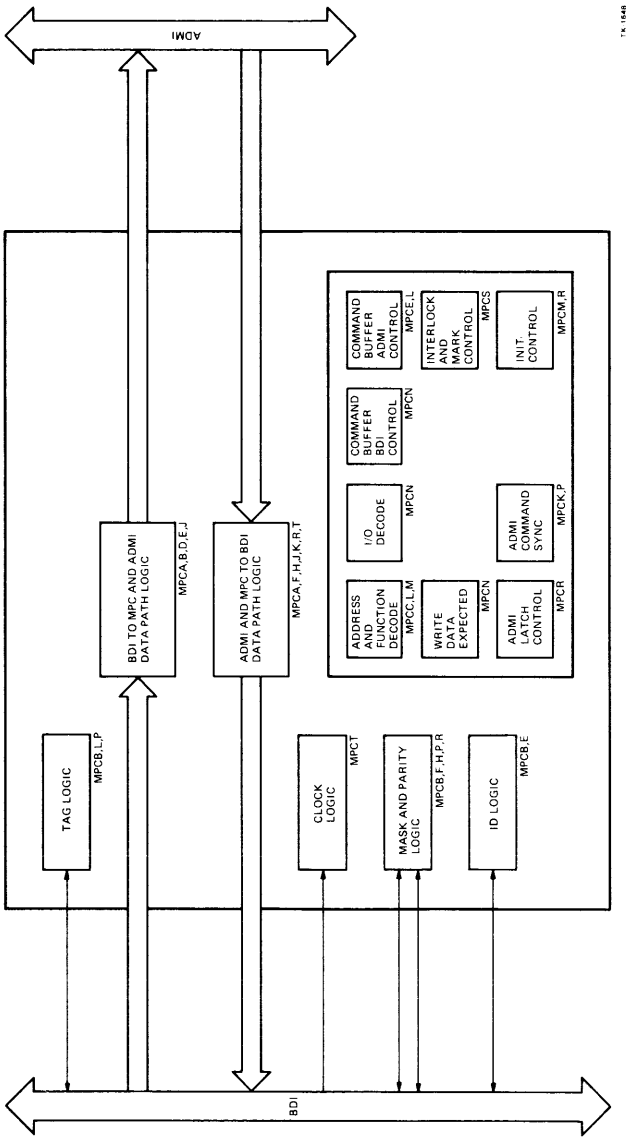
As Used in MS78Ø Main Memory	As Used in MA78Ø Multiport Memory
BUS FL INF 16 H	BM2 BUS ADMI B16 H
BUS FL INF 17 H	BP2 BUS ADMI B17 H
BUS FL INF 18 H	BR2 BUS ADMI B18 H
BUS FL INF 19 H	BS1 BUS ADMI B19 H
BUS FL INF 2Ø H	BV2 BUS ADMI B2Ø H
BUS FL INF 21 H	CA1 BUS ADMI B21 H
BUS FL INF 22 H	CB2 BUS ADMI B22 H
BUS FL INF 23 H	CE2 BUS ADMI B23 H
BUS FL INF 24 H	CF2 BUS ADMI B24 H
BUS FL INF 25 H	CH2 BUS ADMI B25 H
BUS FL INF 26 H	CJ2 BUS ADMI B26 H
BUS FL INF 27 H	CK2 BUS ADMI B27 H
BUS FL INF 28 H	CL2 BUS ADMI B28 H
BUS FL INF 29 H	CM2 BUS ADMI B29 H
BUS FL INF 3Ø H	CR1 BUS ADMI B3Ø H
BUS FL INF 31 H	CR2 BUS ADMI B31 H
BUS FL MSK Ø H	DM2 BUS ADMI MSK Ø H
BUS FL MSK 1 H	DL1 BUS ADMI MSK 1 H
BUS FL MSK 2 H	DK2 BUS ADMI MSK 2 H
BUS FL MSK 3 H	DL2 BUS ADMI MSK 3 H
BUS X PAR 1 H	DK1 BUS ADMI P1 H


SBI TO CPU A
SBI TO CPU B
SBI TO CPU C
SBI TO CPU D
M8258 MULTIPORT INTERFACE MODULE (MPI) (RESIDES IN CPU A CABINET)
M8258 MULTIPORT INTERFACE MODULE (MPI) (RESIDES IN CPU B CABINET)
M8258 MULTIPORT INTERFACE MODULE (MPI) (RESIDES IN CPU C CABINET)
M8258 MULTIPORT INTERFACE MODULE (MPI) (RESIDES IN CPU D CABINET)
BIDIRECTIONAL INTERCONNECT (BDI)
13 COMMAND AND STATUS LINES
15 COMMAND AND STATUS LINES
43 BIDIRECTIONAL LINES (FOR COMMAND/ADDRESS, MASK, ID, AND PARITY)
15 FEET OF CABLE SEPARATES THESE MODULES
M8259 MULTIPORT CONTROLLER MODULE (MPC)
M8259 MULTIPORT CONTROLLER MODULE (MPC)
M8259 MULTIPORT CONTROLLER MODULE (MPC)
ADDRESS/DATA /MASK INTERCONNECT
4 INFORMATION LINES
25 COMMAND LINES
38 ADDRESS/DATA LINES
CONTROL PANEL
MEMORY CONTROL SECTION
M8212 DATA PATH/ECC MODULE (MDT)
DATA PATH TIMING
TIMING INPUTS
ECC DIAGNOSTIC CONTROL
STATUS
M8260 MULTIPORT ARRAY TIMING AND CONTROL MODULE (MAT)
CONTROL INPUTS
CONTROL OUTPUTS
M8261 MULTIPORT SYNCHRONIZER MODULE (MPS)
PRESENCE BIT
72 DATA LINES
6 CONTROL LINES
M8210 INVALIDATE MAP (OPTIONAL)
15 ADDRESS LINES
1 CONTROL LINE
MEMORY ARRAY INTERCONNECT (MAI)
19 ADDRESS LINES
72 DATA LINES
9 CONTROL LINES
8 PRESENCE BIT LINES (1 PER BOARD)
M8210 ARRAY CARD NO. 1
M8210 ARRAY CARD NO. 2
M8210 ARRAY CARD NO. 8
NOTES:
1. ALL MODULES EXCEPT M8258 MULTIPORT INTERFACE MODULES (IN CPU CABINET) PLUG INTO MULTIPORT CONTROLLER BACKPLANE.
2. EACH M8210 ARRAY CARD STORES 32,768 72 BIT WORDS (256 K BYTES).
3. MULTIPORT CONTROLLER CAN ACCOMMODATE UP TO 8 M8210 MEMORY ARRAY CARDS, PROVIDING A TOTAL MAXIMUM STORAGE CAPACITY OF 2 MEGABYTES.
TK 1641

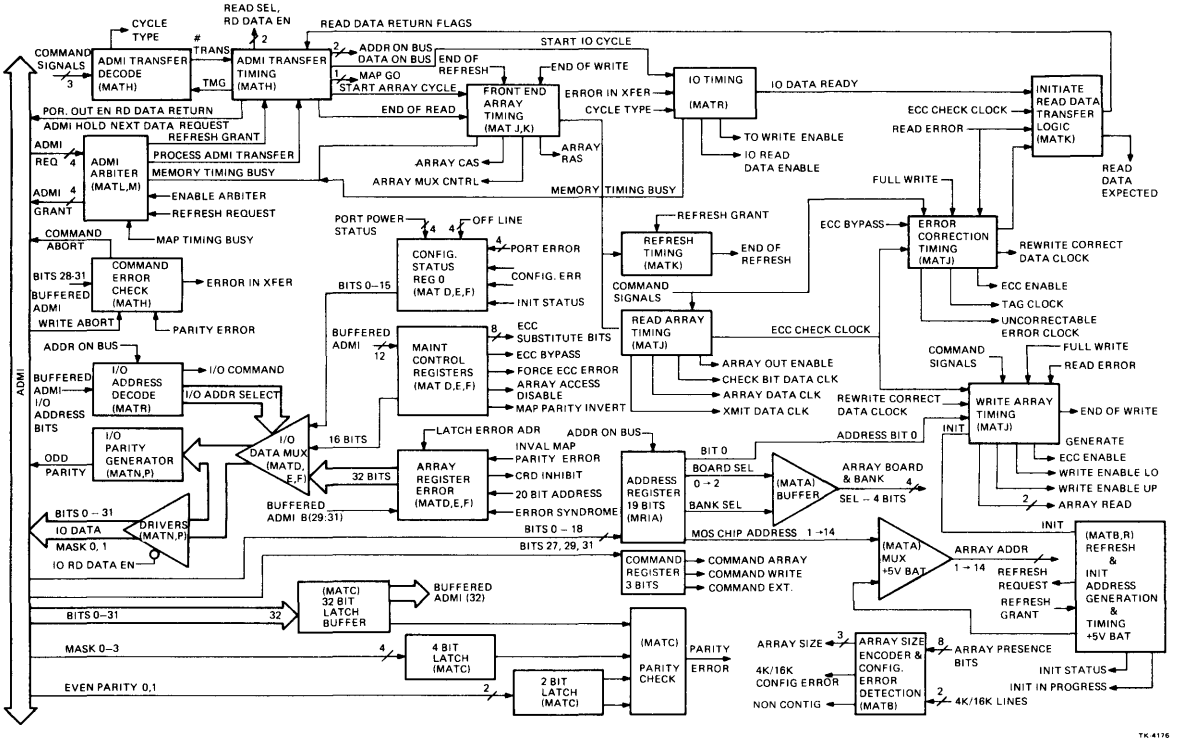
MA780 MULTIPORT MEMORY INTERFACE MODULE (M8258) BLOCK DIAGRAM



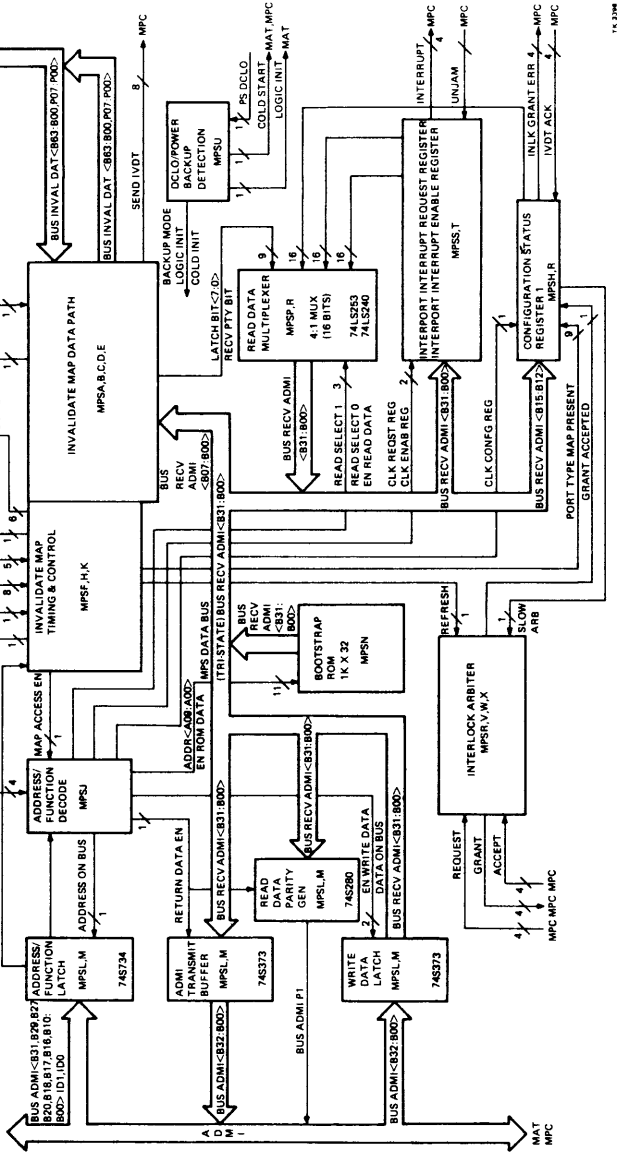
MA780 MULTIPORT MEMORY CONTROLLER MODULE (M8259) BLOCK DIAGRAM



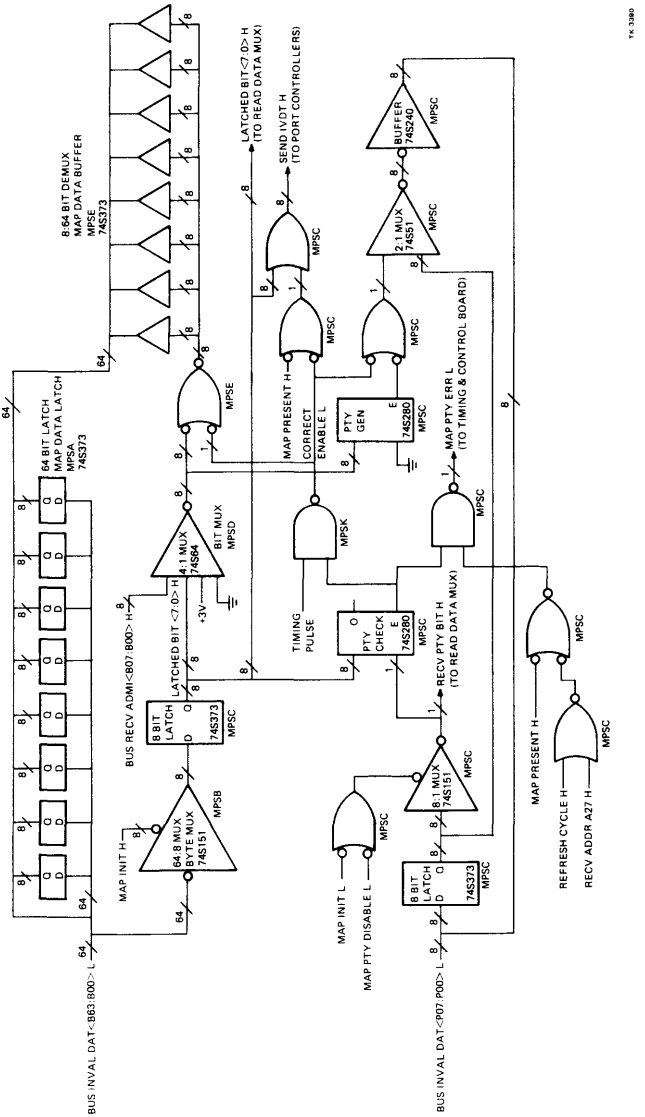
MA780 MULTIPORT MEMORY ARRAY TIMING AND CONTROL MODULE (M8260) BLOCK DIAGRAM



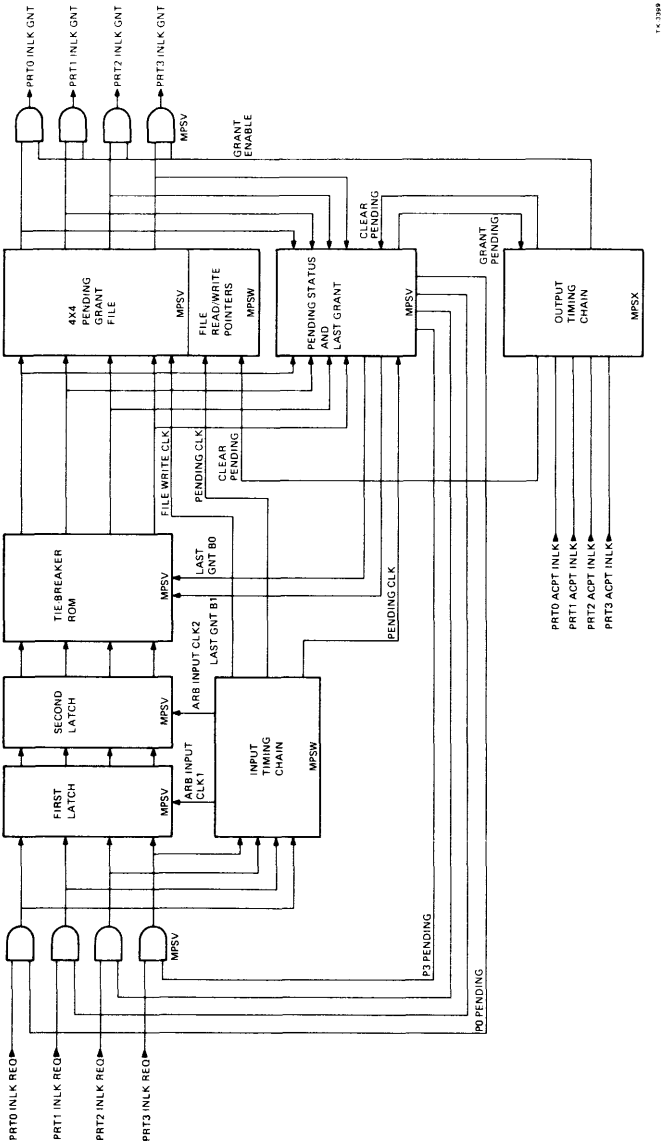
MA780 MULTIPORT MEMORY SYNCHRONIZER MODULE (M8261) BLOCK DIAGRAM



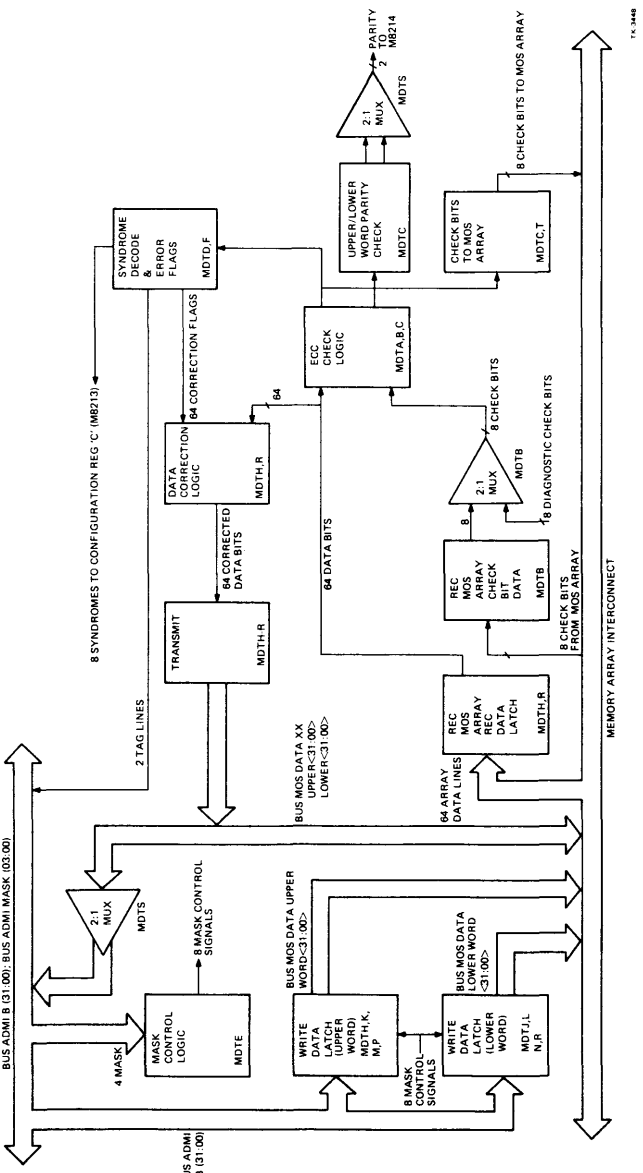
MA780 MULTIPORT MEMORY INVALIDATE MAP DATA PATH BLOCK DIAGRAM



MA780 MULTIPORT MEMORY INTERLOCK ARBITER BLOCK DIAGRAM

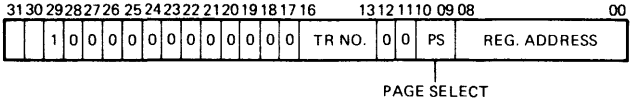


MA780 MULTIPORT MEMORY DATA PATH/ECC MODULE (M8212) BLOCK DIAGRAM



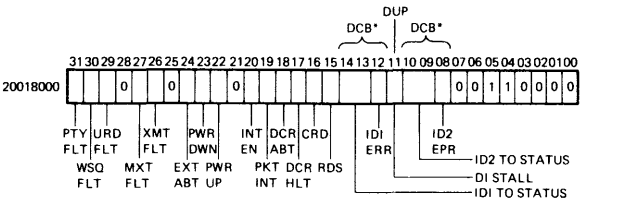
DR780 REGISTERS

DCR ADDRESS REGISTER



TK-4608

DCR READ REGISTER



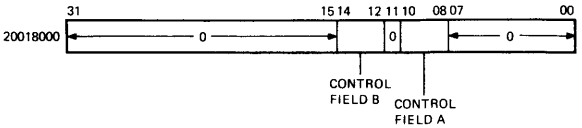
BIT	FUNCTION
31	PARITY FAULT
30	WRITE SEQUENCE FAULT
29	UNEXPECTED READ DATA
28	UNUSED
27	MULTIPLE TRANSMITTER FAULT
26	TRANSMITTER DURING FAULT
25	UNUSED
24	EXTERNAL ABORT
23	POWER DOWN
22	POWER UP
21	UNUSED
20	INTERRUPT ENABLE

*DCB = CONTROL CODE

BIT	FUNCTION
19	PACKET INTERRUPT
18	ABORT
17	HALT
16	CORRECTED READ DATA
15	READ DATA SUBSTITUTE
14	COMMAND/ADDRESS TIME OUT ID1
13	READ DATA EXPECTED TIME OUT ID1
12	RECIEVED ERROR CONFIRMATION ID1
11	DATA INTERCONNECT STALL
10	COMMAND/ADDRESS TIME OUT ID2
9	READ DATA EXPECTED TIME OUT ID2
8	RECIEVED ERROR CONFIRMATION ID2
7 → 0	$30_{(16)}$ ADAPTER TYPE CODE

DR780 REGISTERS (CONT)

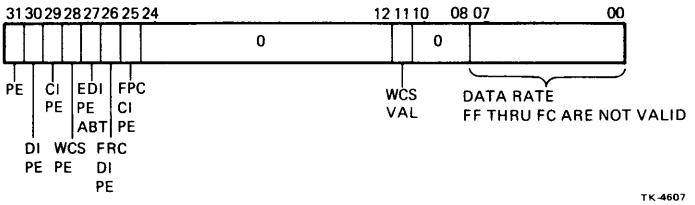
DCR WRITE REGISTER



14	13	12		10	9	8	
0	0	0	NO OPERATION	0	0	0	NO OPERATION
0	0	1	CLEAR CORRECTED READ DATA	0	0	0	CLEAR POWER UP
0	1	0	SET EXTERNAL ABORT	0	1	0	CLEAR POWER DOWN
0	1	1	CLEAR PACKET INTERRUPT	0	1	1	NO OPERATION
1	0	0	RESET	1	0	0	CLEAR ABORT INTERRUPT AND READ DATA SUBSTITUTE
1	0	1	SET OUT OF SEQUENCE TEST	1	0	1	CLEAR INTERRUPT ENABLE
1	1	0	CLEAR OUT OF SEQUENCE TEST	1	1	0	SET INTERRUPT ENABLE
1	1	1	NO OPERATION	1	1	1	CLEAR HALT

TK-4617

DR780 UTILITY REGISTER



TK-4607

DR780 TR ARBITRATION JUMPER AND WIREWRAP SELECTION

Signal Name	TR SELD L	TR SELC L	TR SELB L	TR SELA L	Wirewrap F02L2 To
TR No.	W4	W3	W2	W1	
1	--	--	--	--	F02C1
2	--	--	--	I	F02D1
3	--	--	I	--	F02E1
4	--	--	I	I	F02F2
5	--	I	--	--	F02H2
6	--	I	--	I	F02J1
7	--	I	I	--	F02J2
8	--	I	I	I	F02M1
9	I	--	--	--	F02N1
10	I	--	--	I	F02P1
11	I	--	I	--	F02P2
12	I	--	I	I	F02S2
13	I	I	--	--	F02T2
14	I	I	--	I	F02U1
15	I	I	I	--	F02U2
16	I	I	I	I	--

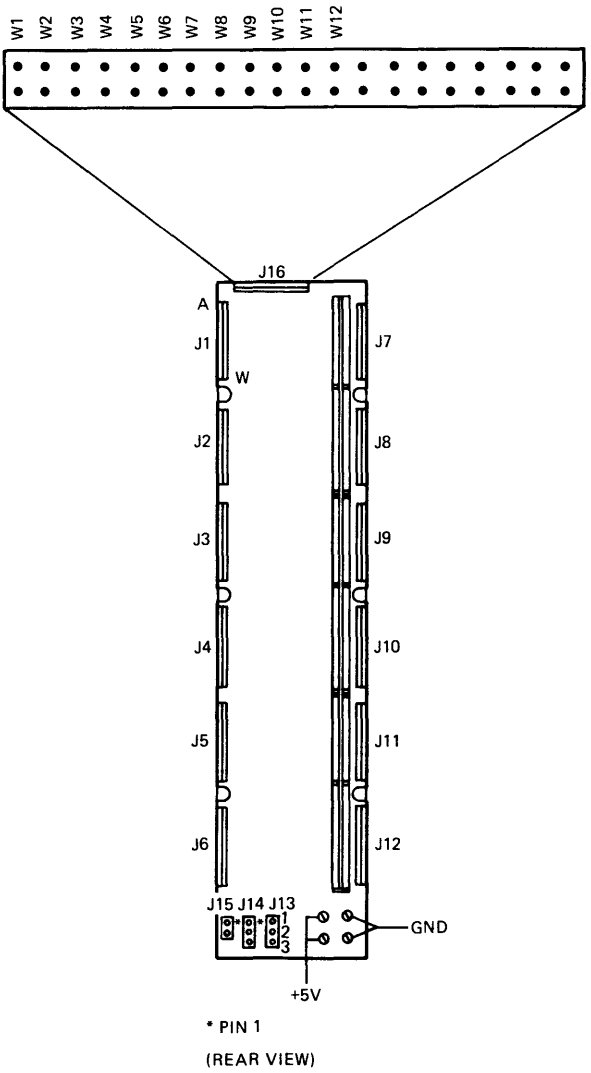
TR Level Jumper - TR arbitration level jumpers for the first DR780 are W4 and W3, for a TR number of 13. TR arbitration level jumpers for the second DR780 are W4, W3, and W1, for a TR number of 14.

TR Wirewrap - Wirewrap BUS SBI TRXX L for the first DR780 from F02L2 to F02T2. Wirewrap BUS SBI TRXX L for the second DR780 from F02L2 to F02U1.

DI Clock Jumper Select - If the DR780 is to be the clock source for the DI, then jumper W8 on the backpanel should be installed anywhere from W9 through W12 (these jumpers not used by the DR780). If the customer's device is to be the source of the DDI clock, then install the jumper on W8.

MSEL Jumper Select - Install the jumper at W7 if the DR780 is not going to perform DDI arbitration, or be its master. If the DR780 is to be the master device, the jumper should be installed on any pins from W9 through W12.

DR780 BACKPLANE



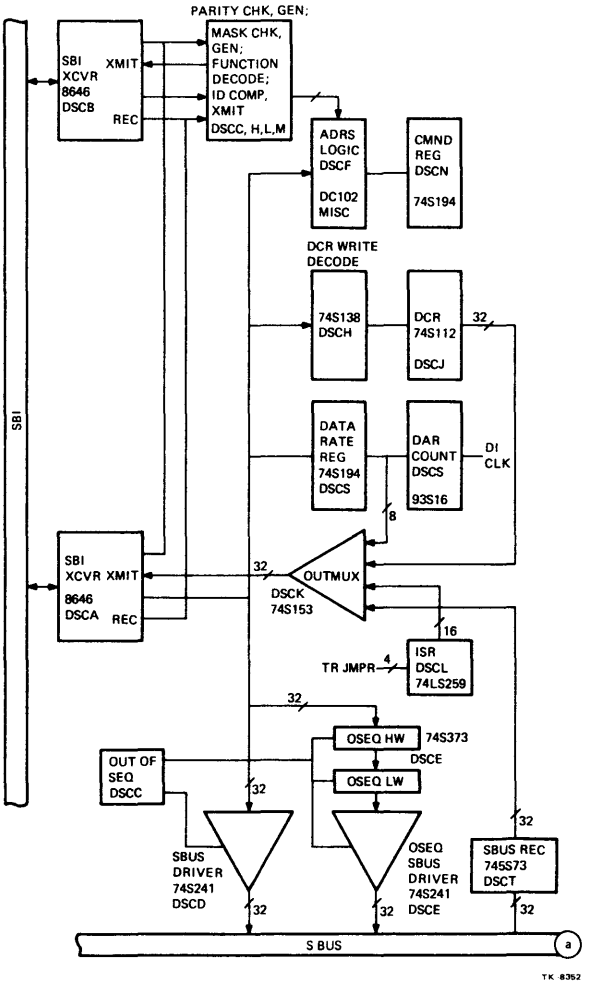
DR780 BACKPLANE (CONT)

	1	2	3	4	5	6
	M8296	M8297	M8298	M8299		M9046
A	DSC	DCB	DUP	DSM		DDIP
B						
C						
D						
E						
F						

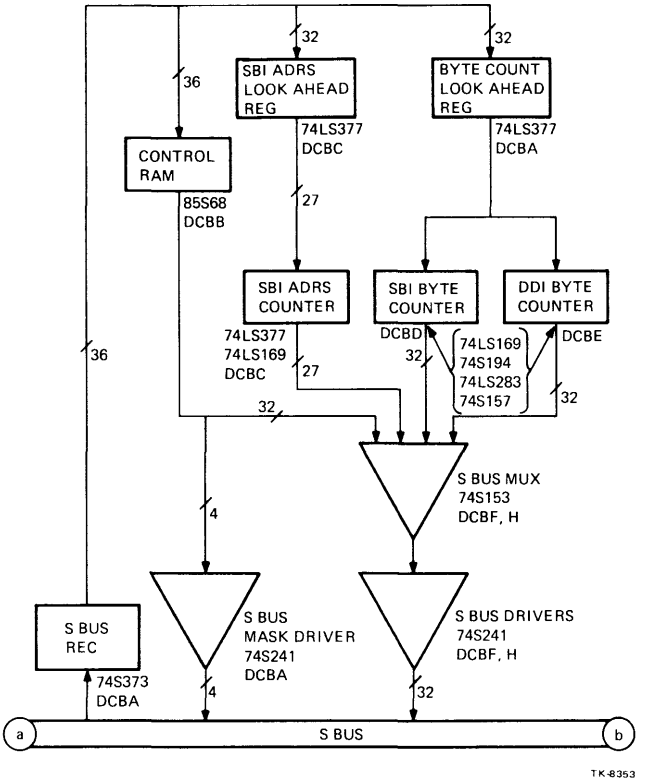
(VIEW FROM SIDE 2)

TK-8348

SBI CONTROL (DSC) M8296

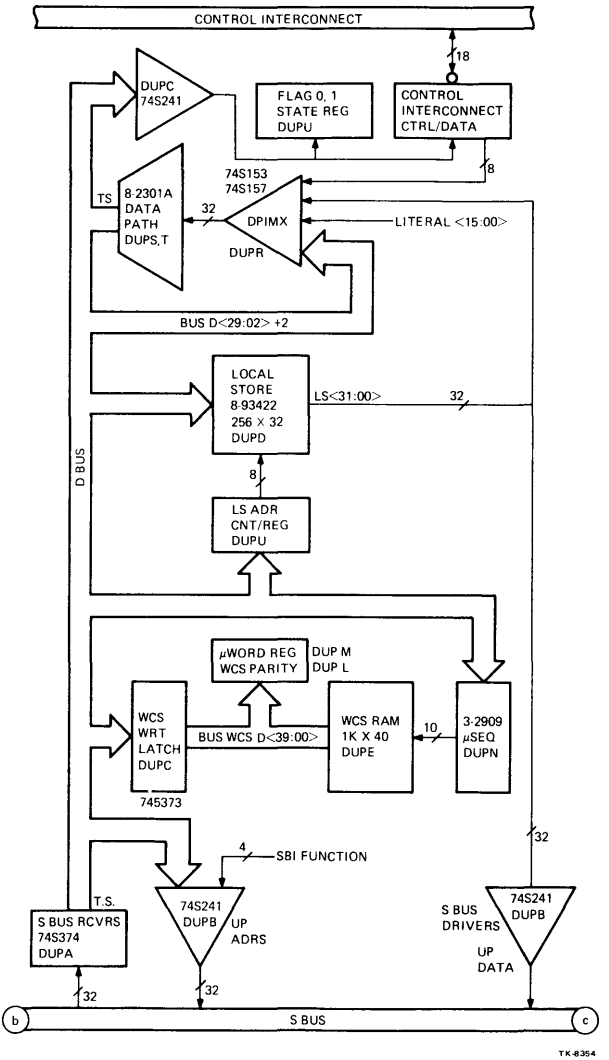


CONTROL BOARD (DCB) M8297

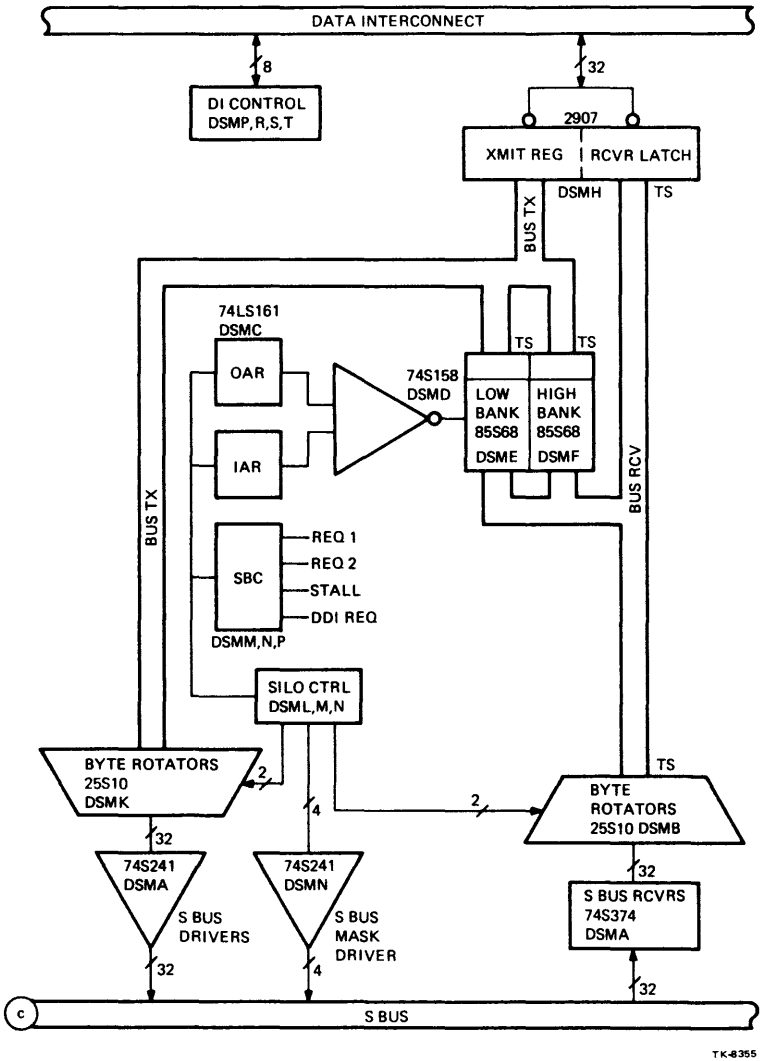


DR780 BLOCK DIAGRAMS (CONT)

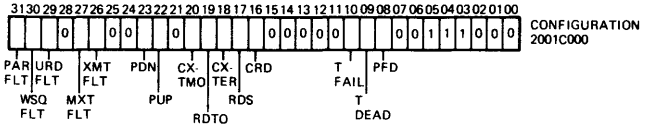
MICROPROCESSOR (DUP) M8298



SILO MODULE (DSM) M8299

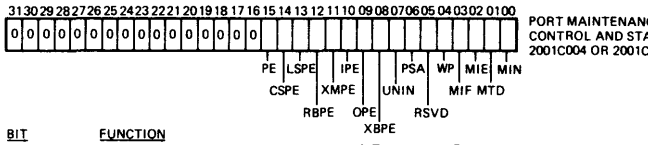


CI780 REGISTERS



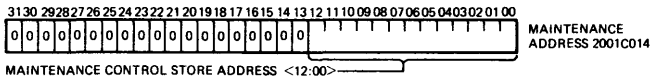
BIT	FUNCTION
31	PARITY FAULT
30	WRITE SEQUENCE FAULT
29	UNEXPECTED READ DATA FAULT
27	MULTIPLE TRANSMITTER FAULT
26	TRANSMIT FAULT
23	ADAPTER POWER DOWN
22	ADAPTER POWER UP

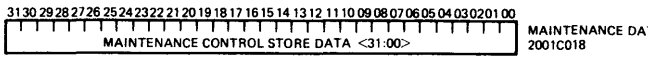
BIT	FUNCTION
20	COMMAND TRANSMIT TIMEOUT
19	READ DATA TIMEOUT
18	COMMAND TRANSMIT ERROR
17	READ DATA SUBSTITUTE
16	CORRECTED READ DATA
10	TRANSMIT FAIL
09	TRANSMIT DEAD
08	POWER FAIL DISABLE
07:00	ADAPTER CODE (38 HEX)



BIT	FUNCTION
15	PARITY ERROR
14	CONTROL STORE PARITY ERROR
13	LOCAL STORE PARITY ERROR
12	RECEIVE BUFFER PARITY ERROR
11	TRANSMIT MULTIPLE PARITY ERROR
10	INPUT PARITY ERROR
09	OUTPUT PARITY ERROR
08	TRANSMIT BUFFER PARITY ERROR

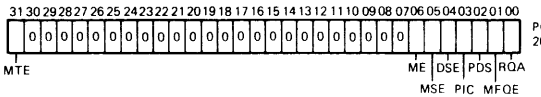
BIT	FUNCTION
07	UNINITIALIZED STATE
06	PROGRAMMABLE STARTING ADDRESS
05	RESERVED
04	WRONG PARITY
03	MAINTENANCE INTERRUPT FLAG
02	MAINTENANCE INTERRUPT ENABLE
01	MAINTENANCE TIMER DISABLE
00	MAINTENANCE INITIALIZE



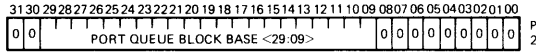


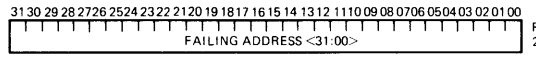
NOTES:
1. ADDRESSES SHOWN ARE FOR A CI780 AT TR14.

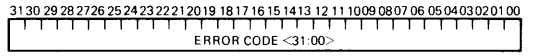
TK-8538

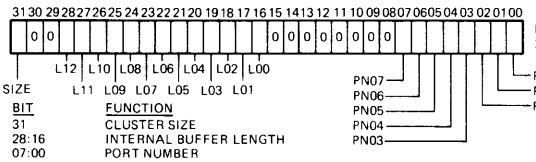


BIT	FUNCTION
31	MAINTENANCE ERROR
06	MAINTENANCE TIMER EXPIRATION
05	MEMORY SYSTEM ERROR
04	DATA STRUCTURE ERROR
03	PORT INITIALIZATION COMPLETE
02	PORT DISABLE COMPLETE
01	MESSAGE FREE QUEUE EMPTY
00	RESPONSE QUEUE AVAILABLE

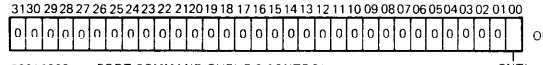








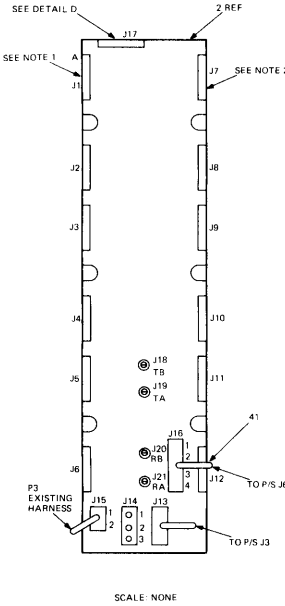
BIT	FUNCTION
31	CLUSTER SIZE
28:16	INTERNAL BUFFER LENGTH
07:00	PORT NUMBER



2001C908	PORT COMMAND QUEUE 0 CONTROL
2001C90C	PORT COMMAND QUEUE 1 CONTROL
2001C910	PORT COMMAND QUEUE 2 CONTROL
2001C914	PORT COMMAND QUEUE 3 CONTROL
2001C918	PORT STATUS RELEASE CONTROL
2001C91C	PORT ENABLE CONTROL
2001C920	PORT DISABLE CONTROL
2001C924	PORT INITIALIZE CONTROL
2001C928	PORT DATAGRAM FREE QUEUE CONTROL
2001C92C	PORT MESSAGE FREE QUEUE CONTROL
2001C930	PORT MAINTENANCE TIMER CONTROL
2001C934	PORT MAINTENANCE TIMER EXPIRATION CONTROL

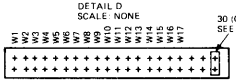
TK-8539

CI780 BACKPLANE JUMPERS



SCALE: NONE

DETAIL D
SCALE: NONE



BOOT TIMER				
W9	W13	W10	W12	TIME
0	0	0	0	0000
0	0	0	1	0100
0	0	1	0	0200
0	0	1	1	0300
0	1	0	0	0400
0	1	0	1	0500
0	1	1	0	0600
0	1	1	1	0700
1	0	0	0	0800
1	0	0	1	0900
1	0	1	0	1000
1	0	1	1	1100
1	1	0	0	1200
1	1	0	1	1300
1	1	1	0	1400
1	1	1	1	1500

SECONDS

0 = JUMPER IN
1 = JUMPER OUT

INTERRUPT PRIORITY LEVEL		
W6	W5	LEVEL
0	0	4
0	1	5
1	0	6
1	1	7

0 = JUMPER OUT
1 = JUMPER IN

TR LEVEL				
W2	W3	W7	W4	TR LEVEL
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14
1	1	1	0	15

0 = JUMPER OUT
1 = JUMPER IN

THE SELECTED TR LEVEL MUST MATCH THE LEVEL CHOSEN BY THE CI780 TR ARBITRATION JUMPER.

CI780 TR ARBITRATION					
FOR	WRAP				
TR1	C01-57 TO C01-53 (BUS TR L)				
TR2	C01-63				
TR3	C01-62				
TR4	C01-65				
TR5	C01-69				
TR6	C01-71				
TR7	C01-73				
TR8	C01-75				
TR9	C01-77				
TR10	C01-81				
TR11	C01-83				
TR12	C01-85				
TR13	C01-86				
TR14	C01-87				
TR15	C01-88				

NOTES:
1. THE SELECTED TR LEVEL MUST MATCH THE TR LEVEL SELECTED BY W2, W3, W4, W7.
2. TR 0 IS RESERVED AS THE HOLD LINE.

CI780 JUMPERS	
J17	SIGNAL
W1	LT JMPR
W2	TR JMPR D
W3	TR JMPR C
W4	TR JMPR A
W5	PRI JMPR 0
W6	PRI JMPR 1
W7	TR JMPR B
W8	PM JMPR
W9	BOOT JMPR 3 H
W10	BOOT JMPR 4 H
W11	RESERVED H
W12	BOOT JMPR 0 H
W13	BOOT JMPR 2 H
W14	DISABLE ARB
W15	EXTEND HDR/TRLR
W16	ALT DELTA TIME
W17	EXTEND ACK TIMEOUT
	OTHER WRAP

2080 JMPR 805-05/805-06
THE STANDARD CONFIGURATION IS TR 14 (W2, W3, W4 IN) AND ALL OTHER JUMPERS OUT.

CI780 BACKPLANE JUMPERS

LT JUMPER (W1)
IN = 2048 SBI CYCLE BEFORE CXTMO.
OUT = 512 SBI CYCLE BEFORE CXTMO.
PANIC MODE JUMPER (W8)
IN = PANIC MODE DISABLED
OUT = PANIC MODE ENABLED

W11 IS RESERVED

DISABLE ARBITRATION (W14)
IN = NO ARBITRATION ON CI BEFORE TRANSMISSION
OUT = NORMAL CI ARBITRATION
EXTEND HEADER/TRAILER (W15)
IN = EXTENDS HEADER/TRAILER
OUT = NORMAL HEADER/TRAILER
(IF W15 IS IN, W17 MUST ALSO BE IN)
ALTER DELTA TIME (W16)
IN = LONG DELTA TIME
OUT = SHORT DELTA TIME
EXTEND ACK TIMEOUT (W17)
IN = LONG TIME OUT
OUT = SHORT TIME OUT

BACKPLANE
MATE-N-LOK CONNECTIONS

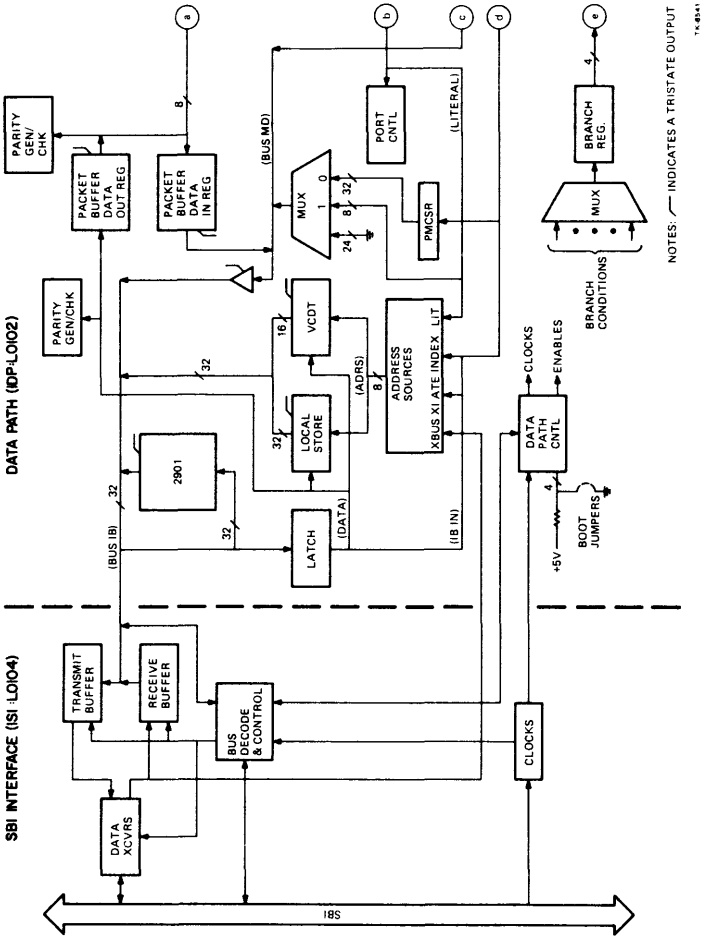
BACKPLANE CONNECTOR	SIGNAL NAME: PS AC LO L	PS DC LO L	-5V	GND
	PIN			
J15	NC	NC	1	2
J14	1	2	NC	3
J13	1	2	NC	3
J16	NC	NC	1+2	3+4

NC = NO CONNECTION

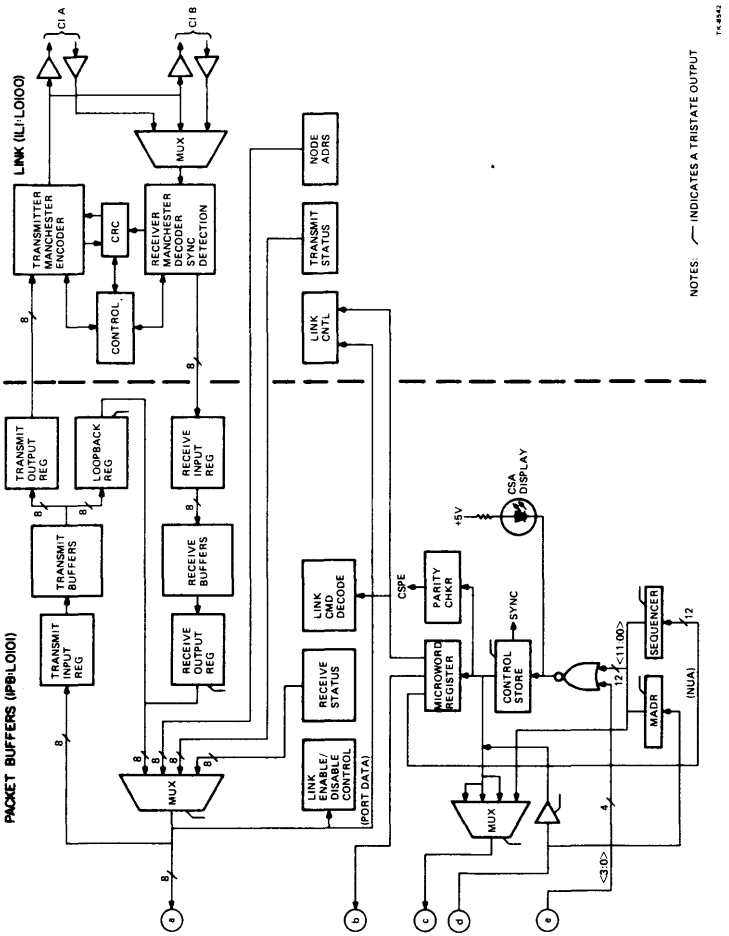
NOTES:
1. CONN J1-J6 ARE SBI BUS OUT CONN.
2. CONN J7-J12 ARE SBI BUS IN CONN.
3. EXTRA JUMPERS, ITEM 30, TO BE SHIPPED IN ITEM 42, PLASTIC BAG.

TK-6540

CI780 BLOCK DIAGRAM, PART 1



CI780 BLOCK DIAGRAM, PART 2



CHAPTER 8

PROCESSOR-SPECIFIC DIAGNOSTICS

Command/Flag	Description
DIAGNOSE	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 the 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 of section has been reached.

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

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

NOTE

In the above to variations of the "/TEST" and "/SECTION" qualifiers, the value of <N> must be less than or equal to <M>. If <M> 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:2F

Dispatch to test number 2F and execute it indefinitely.

DIAG/SECT:B

Dispatch to section number B and execute it indefinitely.

DIAG/PASS:-1

Execute all of the micro diagnostics indefinitely.

	DIAG/TEST:2F/CONT Dispatch to test 2F and start execution of the remaining tests.
CONTINUE	Continues microdiagnostic execution without changing the program control flags.
Set and Clear Flags	
SET/CLEAR FLAG HD	Sets (or clears) the Halt on Error Detection flag.
SET/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 SOMM	Sets (or clears) the Stop on Micro Match bit.
SET/CLEAR SOMM:<ADDRESS>	Loads address into Micromatch Register and sets (or clears) the stop on Micromatch bit.
SET/CLEAR FP:<ADDRESS>	Loads <ADDRESS> into the FPA micro sync register.
SET STEP STATE	Sets the CPU clock to single time state.
SET STEP BUS	Sets the CPU clock to single bus cycle.
	Both the SET STEP 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 hardcore tests are invoked, the current value of the Test 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 NORMAL	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. ID Bus registers T1-T8 are destroyed during the examines, except for the ID Bus and VBus examines. All of the following examines, except V Bus, advance the clock to CPT0 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 Ø 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 SBI:<ADDRESS>	Displays the contents of the SBI 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. (Do not use ID address when examining D register; use EXAMINE DR command.)
EXAMINE QR	Displays the contents of the Q Register.

EXAMINE SC	Displays the contents of the SC register.
EXAMINE FE	Displays the contents of the FE 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 FE: <DATA>

DEPOSIT VA: <DATA>

DEPOSIT PA: <DATA>

DEPOSIT SBI: <DATA>

REPEAT <COMMAND STRING>

BLKMIC

```
BLKMIC  <SCR ADDRESS>, [SCR INDEX], <WCS ADDRESS>,
        <WORD COUNT>, [<WCS ADDRESS INDEX>]
```

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

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

For example, if the current value of the index is 2, 14_8 (<SCR INDEX> * 6) would be added to the <SCR ADDRESS> to find the first 96-bit microword to load into the WCS.

CHKPNT

```
CHKPNT [<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>.

CMPCA

```
CMPCA [<MODE>], <REGISTER>, <DST ADDRESS>, [<DST ADDRESS
      INDEX>]
```

Compares the contents of the console register specified by <REGISTER> with the contents of the location specified by <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 IDREGLO or IDREGHI, the register used in the compare is the ID Bus register that was read in the most recent READID instruction.

CMPCAD

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

Compare by the contents of the console registers specified by <REGISTER> and <REGISTER>+2 with the contents of the registers specified by <DST ADDRESS> and <DST ADDRESS>+2, 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 IDREGLO or IDREGHI, the register used in the compare is the ID Bus register that was read in the most recent READID instruction.

CMPCAM

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

Take the contents of the console register specified by <REGISTER>, 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 IDREGLO or IDREGHI, the register used in the compare is the ID Bus register that was read in the most recent READIN instruction.

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

CMPCMD

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

Take the contents of the console registers specified by <REGISTER> and <REGISTER>+2, mask it with the contents of <MASK ADDRESS> and <MASK ADDRESS>+2, indexed by <MASK ADDRESS INDEX>, and compare it with the contents of <DST ADDRESS> and <DST ADDRESS>+2, 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 IDREGLO or IDREGHI, the register used in the compare is the ID Bus register that was read in the most recent READIN instruction.

The mask is performed by taking the contents of <MASK ADDRESS> and <MASK ADDRESS>+2, indexed by <MASK ADDRESS INDEX>, complementing it, and bit clearing the contents of <REGISTER> and <REGISTER>+2.

CMPPCSV

CMPPCSV <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 last value (specified in the LOOP instruction). If the current value is less than or equal to the last 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 or Error flag is set (ref. subsection 4.6), execution is restarted at this saved address after the IFERROR instruction is executed.

FETCH

FETCH <WCS ADDRESS>, [<WCS ADDRESS INDEX>], [<WCS ROM NOP>]

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 NOP> is specified, clear bit > of the MCR register during the maintenance return.

FLTONE

FLTONE <DST ADDRESS>, <INDEX NAME>

Generate a 32-bit word of all zeros. Insert a logic one in the bit postion 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.

FLTZRO

FLTZRO <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 [<MESSAGE NUMBER>], [<FAIL ADDRESS>]

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

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

INITIALIZE

INITIALIZE

Set and clear the CPU Initialize bit in the Machine Control register, clear the single time state bit, set the single bus cycle bit, set the ROM NOP bit, and set the Proceed bit in the Machine Control register.

KMXGEN

KMXGEN <SRC ADDRESS>, <INDEX NAME>

Generate the KMUX address specified by the current value minus one of <INDEX NAME> and load it into the KMUX field of the microinstruction specified by <SRC ADDRESS>.

LDIDREG

LDIDREG <REGISTER>, <SRC ADDRESS>, [<SRC ADDRESS INDEX>]

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

If <REGISTER> is the microstack, microbreak, or WCS address, the contents of <SCR 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>, [<SIZE 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 with 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 an <INDEX NAME>, save the current value of that index name as the <END> value of this index name.

If <SIZE DEPENDENT> is specified, divide the larger of <START> and <END> by two if there is only one WCS module on the system. Otherwise, leave them unchanged.

MASK

MASK <DST 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>.

NEWTST

NEWTST <TEST NAME>, [<TEST DESCRIPTION>], [<LOGIC DESCRIPTION>], [<ERROR DESCRIPTION>], [<SYNC POINT DESCRIPTION>]

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

READIN

READID <REGISTER>

Reads the ID Bus register specified by <REGISTER> and loads the contents of it into locations IDREGLO and IDREGHI.

RESET

RESET

Executes an <LSI-11 reset instruction>.

REPORT

REPORT <MODULE NAME STRING>

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

TSTVB

TSTVB <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 V Bus data just read. The <SRC TABLE> has the following format:

```
1$:  .WORD    <NUMBER OF BITS TO CHECK>
     VBUSG    <CHANNEL NUMBER>, <BIT NUMBER>, <EXPECTED BIT
              VALUE>
       .
2$:  .WORD    <NUMBER OF BITS TO CHECK>
     VBUSG    <CHANNEL NUMBER>, <BIT NUMBER>, <EXPECTED BIT
              VALUE>
       .
       .
       .
```

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

```
TSTVB    1$,I
```

If the current value of the <SRC TABLE ADDRESS INDEX> is 2, and the <SRC TABLE> looks like the above table, the physical <SRC TABLE ADDRESS> would be 2$.

SETPSW

```
SETPSW <DATA>
```

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

SETVEC

```
SETVEC <VECTOR ADDRESS>
```

Set the LSI-11 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.

TYPSIZE

TYPSIZE

Use the contents of location BADDATA to determine the WCS module configuration and type a message and the number of WCS modules that will be tested. If any of the following conditions exist, the test stream is aborted and the NER (No Error Report) flag is set:

a. WCS module count is zero

b. bits 3-0 are nonzero

c. 5th K of WCS is not present

```
;FIELDS ARRANGED ALPHABETICALLY

ACF/=0,2,70,D           ;ACCELERATOR CONTROL
        NOP=0
        SYNC=1
        TRAP=2
        CONTROL=3       ;ACCELLERATOR-DEPENDENT CONTROL FUNCTION

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

ADS/=0,1,47             ;ADDRESS SELECT
        VA=0
        IBA=1


ALU/=0F,4,66,D          ;ALU CONTROL FUNCTIONS
        A-B=00
        A-B.RLOG=01
        A-B-1=02
        INST.DEP=03                     ;INSTRUCTION DEPENDENT
        A+B+1=04
        A+B=05
        A+B.RLOG=06
        ORNOT=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/=0,2,80             ;AMX TO ALU
        LA=0
        RAMX=1
        RAMX.SXT=2                      ;RAMX SIGN EXTENDED ACCORDING TO DT
        RAMX.OXT=3                      ;RAMX ZERO EXTENDED. OXT(L)=0
```

```
BEN/=0,5,72,D           ;BRANCH ENABLE
        NOP=0                       ;NO BRANCH
        Z=1                         ; ALU Z
        ROR=2                       ;LA<1>, PSL<C>, LA<0>
        C31=3                       ; ALU C31, 0
        ACCEL=6                     ;CODE FROM ACCELERATOR
        DATA.TYPE=8                 ;(VAX MODE) *, ASRC+VSRC, ASRC+Q+D
                                    ;   0--NORMAL, 1--QUAD OR DOUBLE
                                    ;   2--FIELD,  3--ADDRESS
        END.DP1=8                   ;(-11 MODE) *, D CLASS, J CLASS+DM27
        IR2-1=9                     ;(VAX MODE) *, IR<2:1>
        PC.MODES=9                  ;(-11 MODE) *, SM47+SM57+DM47+DM57, DST R=PC
        REI=0A                      ;(VAX MODE) MODE.LSS.ASTLVL, *, *
        SRC.PC=CA                   ;(-11 MODE) SRC R=PC
        IB.TEST=0B                  ; 0--TB MISS, 1--ERROR
                                    ; 2--STALL, 3--DATA OK
        MUL=0C                      ;SC.NE.0, D<1:0>
        SIGNS=0D                    ;Q<31>, D.NE.0, D<31>
        INTERRUPT=0E                ;AC LOW, INTERNAL INTERRUPT, INT SEQ
        DECIMAL=0F                  ;0, D BYTE 0 VALID DIGIT, D2-0 NEG SIGN
        UTRAP=10                    ;MICROTRAP DISPATCH VECTOR
        LAST.REF=11                 ;-FPD, NESTED ERROR, LOW TWO BITS:
                                    ; 0--READ INTERLOCK, 1--READ, READ CHK
                                    ; 2--WRITE, 3--READ, WRITE CHK
        EALU=12                     ;EALU N, EALU Z, SC.NEQ.0, SS
        SC=14                       ;SC<9:8>.NE.0, SC.GT.0, SC<9:5>.NE.0
        ALU1-0=15                   ;RLOG EMPTY, ALU<1:0>=0, ALU<1>, ALU<0>
                                    ; (ALU BITS FROM PREVIOUS STATE)
        STATE7-4=16                 ;STATE <7:4>
        STATE3-0=17                 ;STATE <3:0>
        D.BYTES=18                  ;BYTES 3, 2, 1, 0 OF D.NE.0
        D3-0=19                     ;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                 ;-VA<31:30>, -CONSOLE, IS+CM, KERNEL
        TB.TEST=1D                  ;PTE VALID, ALIGNED, QUAD, +
                                    ; 0--TRANSLATION OK, 1--WR CHK AND M=0
                                    ; 2--ACCESS VIOLATION, 3--TB MISS
```

```
BMX/=0,3,82             ;BMX TO ALU
        MASK=0                          ;A 0 IN THE BIT SELECTED BY SC<4:0>
        PC.OR.LB=1                      ;LB UNLESS R=PC, THEN PC
        PACKED.FL=2                     ;PACKED FLOATING
        LB=3
        LC=4
        PC=5
        KMX=6
        RBMX=7                          ;D OR Q
CCK/=0,3,20,D           ;CONDITION CODES
        NOP=0                           ;DEFAULT
        LOAD.UBCC=1                     ;SAMPLE ALU & EALU CONDITIONS
        SET.V=2                         ;FORCE V, NO EFFECT ON N, Z, C
        TST.Z=3                         ;CLR Z IF ALU.NE.0,
                                        ; SET N FROM AMX[UDT]
        ROR=4                           ;SET N & Z FROM ALU, C FROM AMX 00
        N+Z_ALU=5                       ;SET N AND Z FROM ALU[UDT]
        C_AMX0=6                        ;OTHERS UNAFFECTED
        INST.DEP=7

CID/=0,4,42             ;CONSOLE & ID BUS CONTROL IF FS/1
        NOP=1                           ;DEFAULT, ALLOW AUTO IB READ
        ACK=5                           ;SET CONSOLE ACKNOWLEGE FLAG
        CONT=7                          ;CLEAR CONSOLE MODE
        READ.SC=9                       ;READ ID BUS REG SELECTED BY SC
        READ.KMX=0B                     ;READ ID BUS REG SELECTED BY DKMX
        WRITE.SC=0D                     ;WRITE REG SELECTED BY SC
        WRITE.KMX=0F                    ;WRITE REG SELECTED BY DKMX

DK/=0,4,88,D
        NOP=0                           ;DEFAULT, HOLD
        LEFT2=1                         ;DOUBLE SHIFT LEFT
        RIGHT2=2                        ;DOUBLE SHIFT RIGHT
        DIV=4                           ;IF NOT ALU CRY, SHIFT LEFT
                                        ; ELSE LOAD FROM SHF
        LEFT=5                          ;SHIFT LEFT
        RIGHT=6                         ;SHIFT RIGHT
        SHF=8                           ;LOAD SHF MUX, INTEGER FORMAT
        SHF.FL=9                        ;LOAD SHF MUX, UNPACKED FLOATING FORMAT
        ACCEL=0A                        ;LOAD ACCELERATOR DATA FROM DF BUS
        BYTE.SWAP=0B                    ;REFLECT BYTES AROUND BIT 16
```

```
        Q=0C                            ;LOAD Q THRU DAL
        DAL.SC=0D                       ;LOAD DAL[SC]
        DAL.SV=0E                       ;LOAD DAL[SHF VAL]
        CLR=0F                          ;LOAD ZEROS

DT/=0,2,78,D            ;DATA TYPE
                        ;CONTROLS AMX SIGN/ZERO EXTENDER, SHF AMOUNT,
                        ;CONDITION CODE SETTING, AND MEMORY REFERENCES
        LONG=0                          ;DEFAULT
        WORD=1
        BYTE=2
        INST.DEP=3                      ;INSTRUCTION DEPENDENT --
                                        ;ANY OF ABOVE, OR QUAD/DOUBLE
EALU/=0,3,13                            ;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)

EBMX/=0,2,18            ;EBMX TO EALU
        FE=0                            ;DEFAULT
        KMX=1
        AMX.EXP=2
        SHF.VAL=3                       ;SHIFT VALUE

FEK/=0,1,24,D           ;FE REGISTER CONTROL
        NOP=0                           ;DEFAULT, HOLD
        LOAD=1

FS/=0,1,42              ;FUNCTION SELECT FOR 43-46
        MCT=0                           ;ENABLE MEMORY CONTROL
        CID=1                           ;ENABLE ID BUS AND CONSOLE CONTROL

IEK/=0,2,30             ;INTERRUPT AND EXCEPTION ACKNOWLEDGE
        NOP=0
        ISTR=1                          ;STROBE INTERRUPT REQUESTS
        IACK=2                          ;INTERRUPT ACKNOWLEDGE
        EACK=3                          ;EXCEPTION ACKNOWLEDGE
```

```
IBC/=0,4,92.D           ;IBUF CONTROL FUNCTIONS
        NOP=0                           ;DEFAULT
        STOP=1
        FLUSH=2                         ;FLUSH IB AND LOAD IBA
        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=0C                        ;CLEAR BYTE 0 (VAX OPCODE)
        CLR.1=0D                        ;CLEAR BYTE 1 (VAX SPECIFIER)
        CLR.0-3=0E                      ;CLEAR BYTES 0-3 (-11 OP & DATA)
        CLR.1-5.COND=0F                 ;CLEAR BYTES 1-5 CONDITIONALLY
                                        ; IF THERE IS NO SPECIFIER EVALUATION,
                                        ; CLEAR NOTHING.  IF A SELF-CONTAINED
                                        ; SPECIFIER, CLEAR IT.  IF IMMEDIATE,
                                        ; ABSOLUTE, OR DISPLACEMENT, CLEAR THE
                                        ; ISTREAM LITERAL.

ID.ADDR/=0,6,58         ;ID BUS ADDRESS
        IBUF=0                  ;RD     ;SPECIFIER/LITERAL DATA FROM IB
        DAY.TIME=1              ;RD+WR  ;CURRENT TIME OF DAY...
                                        ; MUST READ UNTIL STOPS CHANGING
        SYS.ID=3                ;RD     ;SYSTEM IDENTIFICATION
        RXCS=4                  ;RD+WR  ;CONSOLE RECIEVE CONTROL/STATUS
        RXDB=5                  ;RD     ;CONSOLE RECIEVE DATA BUFFER
                                        ; (TO-ID REGISTER)
        TXCS=6                  ;RD+WR  ;CONSOLE TRANSMIT CONTROL/STATUS
        TXDB=7                  ;WR     ;CONSOLE TRANSMIT DATA BUFFER
                                        ; (FROM-ID REGISTER)
        DQ=8                            ;DATA PATH D/Q REGISTERS (MAINT ONLY)
        NXT.PER=9               ;WR     ;INTERVAL CLOCK NEXT PERIOD REGISTER
        CLK.CS=0A               ;RD+WR  ;INTERVAL CLOCK CONTROL/STATUS
        INTERVAL=0B             ;RD     ;CURRENT INTERVAL COUNT
        CES=0C                  ;RD+WR  ;CPU ERROR/STATUS
        VECTOR=0D               ;RD+WR  ;EXCEPTION & INTERRUPT CONTROL
        SIR=0E                  ;RD+WR  ;SOFTWARE INTERRUPT REGISTER
        PSL=0F                  ;RD+WR  ;PROCESSOR STATUS LONGWORD
        TBUF=10                         ;TRANSLATION BUFFER DATA
        TBER0=12                        ;TB ERROR/STATUS 0
        TBER1=13                        ;TB ERROR/STATUS 1
        ACC.0=14                        ;ACCELERATOR REGISTER #0
```

```
        ACC.1=15                        ;ACCELERATOR REGISTER #1
        ACC.2=16                        ;ACCELERATOR REGISTER #2
        ACC.CS=17                       ;ACCELERATOR CONTROL/STATUS
        SILO=18                         ;NEXT ITEM FROM SBI HISTORY
        SBI.ERR=19                      ;SBI ERROR REGISTER
        TIME.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             ;WR     ;WRITING WCS COUNTS ADDRESS
;ID BUS ADDRESSES CONTINUED.  ADDRESSES 24-3F ARE RAM LOCATIONS

        P0BR=24                         ;PROCESS SPACE 0 BASE REGISTER
        P1BR=25                         ;PROCESS SPACE 1 BASE REGISTER
        SBR=26                          ;SYSTEM SPACE BASE REGISTER
        KSP=28                          ;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
        D.SV=2E
        Q.SV=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                         ;PROCESS 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,+             ;NEXT MICRO WORD ADDRESS, DEFAULT IS THE
                        ;FOLLOWING MICRO WORD
                        ;SYMBOLS ARE DEFINED BY ":"
KMX/=0,6,58             ;CONSTANTS OR # FROM FK
        .8=0                        ;#8 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 (-11 MODE)
        ZERO=6                      ; OR ZEROS (VAX MODE)
        SC=7                        ;SC[9:0] FROM FK
                        ;8 - 3F: CONSTANTS (1 CYCLE SETUP IF ALU IN ARITH MODE)
                              ;DECIMAL VALUE OF CONSTANT
        .14=8                       ;20     (AF,JL,MH)
        .A0=9                       ;160    (AF,JL)
        .34=0A                      ;52     (AF)
        .28=0B                      ;40     (AF)
        .40=0C                      ;64     (AF,JL,MH,TF)
        .50=0D                      ;80     (AF,MH)
        .3000=0E                    ;12288  (JL)
        .EF=0F                      ;239    (JL)
        .80=10                      ;128    (AF,JL,MH,TF)
        .8000=11                    ;-32768 (AF)
        .FF=12                      ;255    (MH,TF)
        .FF00=13                    ;-256   (MH,AF,JL)
        .1E=14                      ;30     (AF)
        .3F=15                      ;63     (MH,AF,TF)
        .7F=16                      ;127
        .7=17                       ;7      (AF,MH)
        .F=18                       ;15     (MH,CM,AF,TF)
        .10=19                      ;16     (MH,AF,JL,TF)
        .FFE8=1A                    ;-24    (MH,TF)
        .FFF0=1B                    ;-16    (CM,JL,TF,MH)
        .FFF8=1C                    ;-8     (CM,TF,MH)
        .20=1D                      ;32     (CM,JL,MH,TF)
        .30=1E                      ;48     (CM,AF,MH,TF)
        .18=1F                      ;24     (MH,AF,TF)
        .3FF=20                     ;1023   (CM)
```

```
        .C=21                           ;12     (CM,JL,TF,MH)
        .D=22                           ;13     (TF)
        .1F=23                          ;31     (AF,JL,MH,TF)
        .1F00=24                        ;7936   (JL,MH)
        .B0=25                          ;176    (MH)
        .E003=26                        ;       (CM)
        .7C=27                          ;124    (AF)
        .FFE0=28                        ;-32    (JL)
        .60=29                          ;96     (TF)
;       SPARE=2A
        .DFCF=2B                        ;?      (JL)
        .FFEF=2C                        ;-17    (AF)
        .FFF1=2D                        ;-15    (AF)
        .19=2E                          ;25     (AF)
        .FFF9=2F                        ;-7     (AF)
; KMX DEFINITION CONTINUED

        .FFFF=30                        ;-1     (MH,JL,TF)
        .88=31                          ;136    (AF)
        .3030=32                        ;?      (TF)
        .F0=33                          ;240    (TF)
        .C0=34                          ;192    (TF,MH)
        .6=35                           ;6      (CM,JL,TF)
        .9=36                           ;9      (CM)
        .FFF6=37                        ;-10    (CM)
        .FFF5=38                        ;-11    (CM)
        .1A=39                          ;26     (CM,AF,TF)
        .24=3A                          ;36     (CM,MH)
        .1B=3B                          ;27     (CM,AF,TF)
        .FFFC=3C                        ;-4     (CM,TF,MH)
        .A=3D                           ;10     (AF,MH)
        .7E=3E                          ;126    (AF,TF)
;       SPARE=3F
MCT/=02,6,42,D                  ;MEMORY CONTROL
        TEST.RCHK=00                    ;TEST TBUF WITH READ CHECK
        MEM.NOP=02                      ;NEITHER CPU NOR IB GETS MEM CYCLE
        TEST.WCHK=04                    ;TEST TBUF WITH WRITE CHECK
        WRITE.V.NOCHK=0A                ;WRITE, INHIBIT TRAPS
        WRITE.V.WCHK=0C                 ;WRITE, NORMAL VARIETY
        LOCKWRITE.V.XCHK=0E             ;INTERLOCK WRITE, VIRTUAL ADDRESS
```

```
        READ.V.RCHK=10                  ;READ, NORMAL VARIETY
        READ.V.NOCHK=12                 ;READ, INHIBIT TRAPS
        READ.V.MCHK=14                  ;READ FOR MODIFY
        READ.V.IBCHK=16                 ;READ, CHECK CONTROLLED BY IBUFFER
        READ.V.NEWPC=18                 ;BEGIN NEW INSTRUCTION STREAM
                                        ; DATA GOES TO INSTRUCTION BUFFER
        LOCKREAD.V.NOCHK=1A             ;INTERLOCK READ, INHIBIT CHECK
        LOCKREAD.V.MCHK=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=2E                  ;INTERLOCK WRITE, PHYSICAL
        READ.P=32                       ;READ, PHYSICAL
        READ.INT.SUM=36                 ;INTERRUPT SUMMARY READ
        LOCKREAD.P=3A                   ;INTERLOCK READ, PHYSICAL
        ALLOW.IB.READ=3E                ;GIVE IB A CYCLE IF IT WANTS ONE

MSC/=0,4,26,D
        NOP=0                           ;DEFAULT
        CHK.CHM=01                      ;CREATE NEW PSL FOR CHM
        CHK.FLT.OPR=02                  ;UTRAP IF ALU<15>=1, ALU<14:7>=0
        CHK.ODD.ADDR=03
        IRD=04                          ;THIS STATE IS INSTRUCTION DECODE
        LOAD.STATE=05
        LOAD.ACC.CC=06                  ;TAKE CONDITION CODES FROM ACCELERATOR
        READ.RLOG=07                    ;(AND POP RLOG STACK)
        CLR.FPD=08                      ;CLEAR PSL<FPD> BIT
        SET.FPD=09                      ;SET SAME
        CLR.NEST.ERR=0A                 ;CLR NESTED ERROR FLAG IN CPU STATUS
        SET.NEST.ERR=0B                 ;SET SAME
        SECOND.REF=0C                   ;OF UNALIGNED DATA REFERENCE
        RETRY.NO.TRAP=0D                ;APPLY SAVED CONTEXT, INHIBIT TRAPS
        RETRY.TRAP=0E                   ;APPLY SAVED CONTEXT TO THIS REF
        INH.CM.ADDR=0F                  ;ALLOW USE OF FULL 32-BIT ADDRESS

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

```
        PC_VA=1
        PC_IBA=2
        VA+4=3                          ;VA_VA+4
        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 INSTR BUFFER

QK/=0,4,51,D
        NOP=0                           ;DEFAULT, HOLD
        LEFT2=1                         ;DOUBLE SHIFT LEFT 2
        RIGHT2=2                        ;DOUBLE SHIFT RIGHT 2
        LEFT=5
        RIGHT=6
        SHF=8                           ;LOAD SHF, INTEGER FORMAT
        SHF.FL=9                        ;LOAD SHF, UNPACKED FLOATING FORMAT
        DEC.CON=0A                      ;DECIMAL CONSTANT = 6'S IN EACH NIBBLE
                                        ; FOR WHICH ALU CRY OUT IS FALSE
        ACCEL=0B                        ;LOAD ACCELERATOR DATA FROM DF BUS
        D=0C
        ID=0E                           ;LOAD ID BUS
        CLR=0F                          ;LOAD ZERO

RAMX/=0,1,77,D          ;DATA PATH MIXER TO AMX
        D=0                             ;DEFAULT
        Q=1
RBMX/=0,1,77            ;DATA PATH MIXER TO BMX.  SAME BIT AS RAMX
        Q=0
        D=1
SCK/=0,1,23,D           ;SC REGISTER CONTROL
        NOP=0                           ;DEFAULT, HOLD
        LOAD=1                          ;LOAD SMX<09:00>

SGN/=0,3,48,D           ;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                    ;SD_ALU<15>, SS_SS.XOR.ALU<15>
        ADD.SUB=6                       ;SD_ALU<15>, SS_SS.XOR.ALU<15>.XOR.IR<1>
        CLR.SD+SS=7                     ;CLEAR BOTH
```

```
SHF/=0,3,85,D           ;ALU SHIFTER CONTROLS
        ALU=0                           ;DEFAULT, SHF_ALU
        LEFT=1                          ;SHF_ALU(L1), INSERT SI CNTL
        RIGHT=2                         ;SHF_ALU(R1), INSERT SI CNTL
        ALU.DT=3                        ;SHF_ALU(DT: L0,L1,L2,L3), INSERT 0
        RIGHT2=4                        ;SHF_ALU(R2), INSERT SI CNTL
        LEFT3=5                         ;SHF_ALU(L3)

SI/=3,3,55,D            ;SHIFT INPUT CONTROLS
                                        ;       SHF     D       Q
                                        ;       ---     -       -
        DIVD=0                          ;       PSL<N>  Q31     ALU C31
        ASHR=1                          ;       ALU 31  Q0      Q31
        ASHL=2                          ;       0       0       D31
        ZERO=3                          ;       0       0       0
        SPARE=4
        DIV=5                           ;       Q31     Q31     ALU C31
        MUL+=6                          ;       0       ALU 0,1 0
        MUL-=7                          ;       1       ALU 0,1 1

SMX/=0,2,16             ;MIXER TO SC
        EALU=0                          ;EALU <9:0>
        FE=1                            ;FE<9:0>
        ALU=2                           ;ALU<09:00>
        ALU.EXP=3                       ;ALU<14:07>

SPO/=0,7,35,D                   ;SCRATCH PAD OPCODE, 7 BITS
        NOP=0                           ;DEFAULT
        LOAD.LC.SC=6                    ;LOAD LC, ADR=SC[03:00]
        WRITE.RC.SC=7                   ;WRITE RC, ADR=SC[03:00]

SPO.AC/=0,4,38          ;4 FUNCTION BITS OF SPO FIELD
        LOAD.LAB=1                      ;LOAD LA, LB FROM R(ACN)
        LOAD.LA=2                       ;LOAD LA_RN, HOLD LB
        WRITE.RAB=3                     ;WRITE RA, RB (ACN)

SPO.ACN/=0,3,35         ;AC NUMBER IN SPO 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

SPO.ACN11/=0,3,35       ;AC NUMBER IN SPO FIELD -- 11 MODE
                                ;-11 MODE       RA              RB
        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>

SPO.R/=0,3,39           ;SCRATCH PAD FUNCS WITH LOW 4 BITS OF SP AS ADR
        LOAD.LC=2                       ;LOAD LC, ADR=SPO.RN
        WRITE.RC=3                      ;WRITE RC
        LOAD.LAB=4                      ;LOAD LA, LB
        WRITE.RAB=5                     ;WRITE RA, RB
        LOAD.LAB1.WRITE.RC=6            ;LOAD LA, LB[R1], AND WRITE RC[RN]
        LOAD.LC.WRITE.RAB1=7            ;LOAD LC[RN], AND WRITE RA, RB[R1]
SPO.RAB/=0,4,35         ;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                   ;R14 = STACK POINTER
        R15=0F                  ;R15 = PC, TO SOFTWARE, SCRATCH TO UCODE
SPO.RC/=0,4,35          ;RC LOCATIONS
        T0=0
        T1=1
        T2=2
        T3=3
```

```
        T4=4
        T5=5
        T6=6
        T7=7
        LC.SV=8                         ;MEM MGMT SAVES LC HERE
        VA.SV=9
        PTE.VA=0A
        PTE.PA=0B
        PC.SV=0C
        SC.SV=0D
        VA.REF=0E
        MBIT.VA=0F
        PTE.MASK=0F

SUB/=0,2,64,D           ;SUBROUTINE CONTROL
        NOP=0                           ;DEFAULT
        CALL=1                          ;PUSH UPC OF THIS MICROINSTRUCTION
                                        ; ONTO USTACK
        RET=2                           ;"OR" TOP OF USTACK TO UPC
                                        ; AND POP USTACK
        SPEC=3                          ;REPLACE LOW 8 BITS OF NEXT
                                        ; UPC WITH SPECIFIER DECODE FROM
                                        ; INSTRUCTION BUFFER

VAK/=0,1,25,D
        NOP=0                           ;DEFAULT
        LOAD=1                          ;LOAD VA


;ALU_0... THRU ALU_D.AND...

ALU_0(A)        "AMX/RAMX.0XT,DT/LONG,ALU/A"
ALU_0[]D        "ALU/@1,AMX/RAMX.0XT,LONG,BMX/RBMX,RBMX/D"
ALU_0-D         "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B"
ALU_0-D-1       "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B-1"
ALU_0+D         "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A+B"
ALU_0-K[]       "AMX/RAMX.0XT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B"
ALU_0-K[]-1     "KMX/@1,BMX/KMX,AMX/RAMX.0XT,DT/LONG,ALU/A-B-1"
ALU_0+K[]       "KMX/@1,BMX/KMX,AMX/RAMX.0XT,DT/LONG,ALU/A+B"
ALU_0+K[]+1     "KMX/@1,BMX/KMX,AMX/RAMX.0XT,DT/LONG,ALU/A+B+1"
```

```
ALU_0+LB+1          "AMX/RAMX.0XT,DT/LONG,BMX/LB,ALU/A+B+1"
ALU_0+LC            "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B"
ALU_0-LC            "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A-B"
ALU_0+LC+1          "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B+1"
ALU_0-LC-1          "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A-B-1"
ALU_0+MASK+1        "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1"
ALU_0+Q             "AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B"
ALU_0-Q             "AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B"
ALU_0-Q-1           "AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B-1"
ALU_0+Q+1           "AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B+1"
ALU_-1              "AMX/RAMX.0XT,DT/LONG,ALU/NOTA"
ALU_D               "RAMX/D,AMX/RAMX,ALU/A"
ALU_D.0XT[]         "RAMX/D,AMX/RAMX.0XT,DT/@1,ALU/A"
ALU_D.0XT[].AND.K[]       "RAMX/D,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/AND"
ALU_D.0XT[].ANDNOT.K[]    "ALU/ANDNOT,AMX/RAMX.0XT,DT/@1,RAMX/D,BMX/KMX,KMX/@2"
ALU_D.0XT[]+K[]  "RAMX/D,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_D.0XT[]-K[]  "RAMX/D,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B"
ALU_D.0XT[]+LC   "ALU/A+B,AMX/RAMX.0XT,DT/@1,RAMX/D,BMX/LC"
ALU_D.0XT[]+Q    "ALU/A+B,AMX/RAMX.0XT,DT/@1,RAMX/D,BMX/RBMX,RBMX/Q"
ALU_D.0XT[]-Q    "RAMX/D,AMX/RAMX.0XT,DT/@1,RBMX/Q,BMX/RBMX,ALU/A-B"
ALU_D.AND.K[]    "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND"
ALU_D.AND.MASK   "RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND"
ALU_D.ANDNOT.K[]          "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_D.ANDNOT.MASK         "RAMX/D,AMX/RAMX,BMX/MASK,ALU/ANDNOT"
ALU_D.ANDNOT.Q   "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT"
;ALU_D(B)... THRU ALU_D.XOR...

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

```
ALU_D.ORNOT.MASK          "RAMX/D,AMX/RAMX,BMX/MASK,ALU/ORNOT"
ALU_D.OR.Q          "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR"
ALU_D[]Q            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1"
ALU_D+Q             "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B"
ALU_D-Q             "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B"
ALU_D+Q+1           "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1"
ALU_D-Q-1           "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B-1"
ALU_D+Q+PSL.C       "ALU/A+B+PSL.C,AMX/RAMX,BMX/RBMX,RBMX/Q,RAMX/D"
ALU_D.SXT[]         "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A"
ALU_D.SXT[].AND.K[]       "RAMX/D,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/AND"
ALU_D.SXT[]+K[]  "RAMX/D,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_D.XOR.K[]       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR"
ALU_D.XOR.LC        "RAMX/D,AMX/RAMX,BMX/LC,ALU/XOR"
ALU_D.XOR.Q         "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/XOR"
ALU_D.XOR.R[]       "RAMX/D,AMX/RAMX,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/LB,ALU/XOR"
ALU_D.XOR.RC[]      "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR"
;ALU_K... THRU ALU_PC...

ALU_K[]             "KMX/@1,BMX/KMX,ALU/B"
ALU_LA              "AMX/LA,ALU/A"
ALU_LA.AND.K[]      "AMX/LA,KMX/@1,BMX/KMX,ALU/AND"
ALU_LA.ANDNOT.K[]         "AMX/LA,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_LA.ANDNOT.MASK        "AMX/LA,BMX/MASK,ALU/ANDNOT"
ALU_LA.XOR.LC       "AMX/LA,BMX/LC,ALU/XOR"
ALU_LA[]D           "AMX/LA,RBMX/D,BMX/RBMX,ALU/@1"
ALU_LA-D            "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B"
ALU_LA-D-1          "AMX/LA,RBMX/D,BMX/RBMX,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,KMX/@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[]LB          "AMX/LA,BMX/LB,ALU/@1"
ALU_LA+LC           "ALU/A+B,AMX/LA,BMX/LC"
ALU_LA[]Q           "AMX/LA,RBMX/Q,BMX/RBMX,ALU/@1"
ALU_LA+Q            "ALU/A+B,AMX/LA,BMX/RBMX,RBMX/Q"
ALU_LA-Q            "ALU/A-B,AMX/LA,BMX/RBMX,RBMX/Q"
ALU_LA-Q-1          "ALU/A-B-1,AMX/LA,BMX/RBMX,RBMX/Q"
ALU_LB              "BMX/LB,ALU/B"
ALU_LC              "BMX/LC,ALU/B"
```

```
ALU_NOT.D          "ALU/NOTA,AMX/RAMX,RAMX/D"
ALU_NOT.RC[]       "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,AMX/RAMX.0XT,DT/LONG,ALU/ORNOT"
ALU_PACK.FP        "BMX/PACKED.FL,ALU/B"
ALU_PC             "BMX/PC,ALU/B"
;ALU_Q... THRU CACHE_...

ALU_Q              "RAMX/Q,AMX/RAMX,ALU/A"
ALU_Q.0XT[]        "RAMX/Q,AMX/RAMX.0XT,DT/@1,ALU/A"
ALU_Q.0XT[].ANDNOT.K[]   "ALU/ANDNOT,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.0XT[].OR.K[]       "ALU/OR,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.0XT[].OR.D         "ALU/OR,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/RBMX,RBMX/D"
ALU_Q.0XT[]+D    "ALU/A+B,AMX/RAMX.0XT,DT/@1,BMX/RBMX,RBMX/D,RAMX/Q"
ALU_Q.0XT[]+D+1  "ALU/A+B+1,AMX/RAMX.0XT,DT/@1,BMX/RBMX,RAMX/Q,RBMX/D"
ALU_Q.0XT[]+K[]  "ALU/A+B,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.0XT[]-K[]  "ALU/A-B,AMX/RAMX.0XT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.AND.K[]    "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND"
ALU_Q.ANDNOT.MASK          "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ANDNOT"
ALU_Q.ANDNOT.K[]           "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_Q(B)           "RBMX/Q,BMX/RBMX,ALU/B"
ALU_Q[]D           "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/@1"
ALU_Q-D            "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B"
ALU_Q-D-1          "ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D"
ALU_Q+K[]          "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B"
ALU_Q-K[]          "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B"
ALU_Q+K[]+1        "ALU/A+B+1,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@1"
ALU_Q+LB           "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B"
ALU_Q-LB           "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A-B"
ALU_Q+LB+1         "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B+1"
ALU_Q+LC           "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B"
ALU_Q-LC           "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B"
ALU_Q+LC+1         "ALU/A+B+1,AMX/RAMX,RAMX/Q,BMX/LC"
ALU_Q+MASK         "ALU/A+B,AMX/RAMX,RAMX/Q,BMX/MASK"
ALU_Q-MASK-1       "ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/MASK"
ALU_Q.OR.K[]       "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR"
ALU_Q.OR.LC        "RAMX/Q,AMX/RAMX,BMX/LC,ALU/OR"
ALU_Q.ORNOT.K[]    "ALU/ORNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@1"
ALU_Q.SXT[]        "ALU/A,AMX/RAMX.SXT,DT/@1,RAMX/Q"
ALU_Q.SXT[].ANDNOT.K[]   "ALU/ANDNOT,AMX/RAMX.SXT,RAMX/Q,BMX/KMX,KMX/@2,DT/@1"
ALU_Q.SXT[]+K[]  "RAMX/Q,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_Q.SXT[]+LB   "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/LB,ALU/A+B"
```

```
ALU_Q.SXT[]+PC   "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/PC,ALU/A+B"
ALU_Q.XOR.D      "RAMX/Q,AMX/RAMX,BMX/RBMX,RBMX/D,ALU/XOR"
ALU_Q.XOR.K[]    "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR"
ALU_Q.XOR.LC     "RAMX/Q,AMX/RAMX,BMX/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,AMX/LA,KMX/@2,BMX/KMX,ALU/AND"
ALU_R[].ANDNOT.MASK      "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/MASK,ALU/ANDNOT"
ALU_R(DST)       "SPO.AC/LOAD.LAB,SPO.ACN11/DST.DST,AMX/LA,ALU/A"
ALU_R[].OR.K[]   "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/OR"
ALU_R[].ORNOT.K[] "ALU/ORNOT,AMX/LA,BMX/KMX,SPO.R/LOAD.LAB,SPO.RAB/@1,KMX/@2"
ALU_R[].XOR.K[]  "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,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,KMX/@1,BMX/KMX,ALU/A+B.RLOG"

CACHE_D[]        "VAK/NOP,MCT/WRITE.V.WCHK,DT/@1,DK/NOP"
CACHE[]_D        "VAK/NOP,MCT/WRITE.V.WCHK,MSC/@1,DK/NOP"
CACHE_D.INST.DEP         "VAK/NOP,MCT/WRITE.V.WCHK,DT/INST.DEP,DK/NOP"
CACHE_D[].NOCHK "VAK/NOP,MCT/WRITE.V.NOCHK,DT/@1,DK/NOP"
CACHE.P_D[]      "VAK/NOP,MCT/WRITE.P,DT/@1,DK/NOP"
CACHE_D[].LK     "VAK/NOP,MCT/LOCKWRITE.V.XCHK,DT/@1,DK/NOP"
;D_0... THRU D_CACHE...

D_0              "DK/CLR"
D_0-D            "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_0+K[]+1        "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_0+LC+1         "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,DK/SHF"
D_0-Q            "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_ACCEL&SYNC     "DK/ACCEL,ACF/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_K[.20]"
D[]_CACHE        "VAK/NOP,MCT/READ.V.RCHK,DT/@1,DK/NOP"
D_CACHE[]        "VAK/NOP,MCT/READ.V.RCHK,MSC/@1,DK/NOP"
D[]_CACHE.IBCHK "VAK/NOP,MCT/READ.V.IBCHK,DT/@1,DK/NOP"
```

```
D_CACHE.INST.DEP        "VAK/NOP,MCT/READ.V.IBCHK,DT/INST.DEP,DK/NOP"
D_CACHE.LK[]     "VAK/NOP,MCT/LOCKREAD.V.WCHK,MSC/@1,DK/NOP"
D[]_CACHE.LK     "VAK/NOP,MCT/LOCKREAD.V.WCHK,DT/@1,DK/NOP"
D[]_CACHE.NOCHK  "VAK/NOP,MCT/READ.V.NOCHK,DT/@1,DK/NOP"
D[]_CACHE.P      "VAK/NOP,MCT/READ.P,DT/@1,DK/NOP"
D[]_CACHE.WCHK   "VAK/NOP,MCT/READ.V.WCHK,DT/@1,DK/NOP"
D_CACHE.WCHK[]   "VAK/NOP,MCT/READ.V.WCHK,MSC/@1,DK/NOP"
;D_DAL... THRU D_D...

D_DAL.NORM       "DK/DAL.SV"
D_DAL.SC         "DK/DAL.SC"
D_D.OXT[]        "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_D.OXT[].ANDNOT.K[]     "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.OXT[]+K[]    "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF"
D_D.OXT[].OR.Q   "RAMX/D,AMX/RAMX.OXT,DT/@1,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,DK/SHF"
D_D.OXT[]+Q      "ALU/A+B,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RBMX/Q,D_ALU"
D_D.OXT[]+Q+1    "RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/RBMX,ALU/A+B+1,D_ALU"
D_D.OXT[].XOR.Q  "DK/SHF,ALU/XOR,SHF/ALU,AMX/RAMX.OXT,RAMX/D,DT/@1,RBMX/Q,BMX/RBMX"
D_D.OXT[].XOR.RC[]       "RAMX/D,AMX/RAMX.OXT,DT/@1,SPO.R/LOAD.LC,SPO.RC/@2,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF"
D_D.AND.K[]      "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_D.AND.K[].LEFT2        "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU.DT,DT/LONG,DK/SHF"
D_D.AND.K[].RIGHT        "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,DK/SHF"
D_D.AND.LC       "RAMX/D,AMX/RAMX,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D_D.AND.MASK     "RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,DK/SHF"
D_D.AND.Q        "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/AND,SHF/ALU,DK/SHF"
D_D.AND.RC[]     "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D_D.ANDNOT.K[]   "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.ANDNOT.LC    "RAMX/D,AMX/RAMX,BMX/LC,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.ANDNOT.PSWZ  "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.4,SHF/ALU"
D_D.ANDNOT.Q     "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.ANDNOT.RC[]  "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D(FRAC)        "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,DK/SHF.FL"
D_D[]K[]         "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/@1,SHF/ALU,DK/SHF"
D_D+K[]          "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF"
D_D-K[]          "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
D_D+K[]+1        "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_D+LB           "RAMX/D,AMX/RAMX,BMX/LB,ALU/A+B,SHF/ALU,DK/SHF"
D_D+LC           "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,DK/SHF"
D_D-LC           "RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,DK/SHF"
D_D+LC+PSL.C     "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+PSL.C,SHF/ALU,DK/SHF"
```

```
D_D.LEFT          "DK/LEFT"
D_D.LEFT2         "DK/LEFT2"
D_D[]MASK         "RAMX/D,AMX/RAMX,BMX/MASK,ALU/@1,SHF/ALU,DK/SHF"
D_D.OR.ASCII      "D_D.OR.K[.30]"
D_D.OR.K[]        "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF"
D_D.ORNOT.MASK    "RAMX/D,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,DK/SHF"
D_D.OR.PSWC       "DK/SHF,ALU/OR,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.1,SHF/ALU"
D_D.OR.PSWV       "DK/SHF,ALU/OR,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.2,SHF/ALU"
D_D.OR.Q          "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,DK/SHF"
D_D.OR.RC[]       "RAMX/D,AMX/RAMX,SPO.P/LOAD.LC.SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,DK/SHF"
D_D[]Q            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1,SHF/ALU,DK/SHF"
D_D+Q             "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF"
D_D-Q             "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_D+Q+1           "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_D-Q-1           "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B-1,SHF/ALU,DK/SHF"
D_D.RIGHT         "DK/RIGHT"
D_D.RIGHT2        "DK/RIGHT2"
D_D.RIGHT(B)      "RBMX/D,BMX/RBMX,ALU/B,SHF/RIGHT,DK/SHF"
D_D.SWAP          "DK/BYTE.SWAP"
D_D.SXT[]         "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_D.SXT[].RIGHT   "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/RIGHT,DK/SHF"
D_D.XOR.K[]       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR,SHF/ALU,DK/SHF"
D_D.XOR.LC        "RAMX/D,AMX/RAMX,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF"
D_D.XOR.Q         "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/XOR,SHF/ALU,DK/SHF"

;D_INT.SUM... THRU D_PC...

D_INT.SUM         "MCT/READ.INT.SUM,DK/NOP"
D_K[]             "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,DK/SHF"
D_K[].RIGHT       "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT,DK/SHF"
D_K[].RIGHT2      "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT2,DK/SHF"
D_LA              "AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_LA.AND.K[]      "AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_LA+D+PSL.C      "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B+PSL.C,SHF/ALU,DK/SHF"
D_LA-D            "DK/SHF,ALU/A-B,AMX/LA,BMX/RBMX,RBMX/D,SHF/ALU"
D_LA(FRAC)        "AMX/LA,ALU/A,SHF/ALU,DK/SHF.FL"
D_LA-K[]          "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
D_LA.RIGHT        "AMX/LA,ALU/A,SHF/RIGHT,DK/SHF"
D_LB              "BMX/LB,ALU/B,SHF/ALU,DK/SHF"
D_LB.PC           "BMX/PC.OR.LB,ALU/B,SHF/ALU,DK/SHF"
```

```
D_LC              "BMX/LC,ALU/B,SHF/ALU.DK/SHF"
D_LC(FRAC)        "BMX/LC,ALU/B,SHF/ALU,DK/SHF.FL"
D_NOT.D           "RAMX/D,AMX/RAMX,ALU/NOTA,SHF/ALU,DK/SHF"
D_NOT.K[]         "KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT,SHF/ALU,DK/SHF"
D_NOT.MASK        "BMX/MASK,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT,SHF/ALU,DK/SHF"
D_NOT.Q           "RAMX/Q,AMX/RAMX,ALU/NOTA,SHF/ALU,DK/SHF"
D_NOT.R[]         "LA_RA[@1],AMX/LA,ALU/NOTA,D_ALU"
D_PACK.FP         "BMX/PACKED.FL,ALU/B,SHF/ALU,DK/SHF"
D_PACK.FP.LEFT    "BMX/PACKED.FL,ALU/B,SHF/LEFT,DK/SHF"
D_PC              "BMX/PC,ALU/B,SHF/ALU,DK/SHF"
D_PC.LEFT         "BMX/PC,ALU/B,SHF/LEFT,DK/SHF"
;D_Q...

D_Q               "DK/Q"
D_Q.OXT[]         "RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_Q.AND.K[]       "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_Q.AND.LC        "RAMX/Q,AMX/RAMX,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D_Q.AND.MASK      "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,DK/SHF"
D_Q.ANDNOT.D      "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_Q.ANDNOT.K[]    "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_Q.ANDNOT.MASK   "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_Q.ANDNOT.PSWC   "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.1,SHF/ALU"
D_Q.ANDNOT.PSWN   "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.8,SHF/ALU"
D_Q.ANDNOT.PSWZ   "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.4,SHF/ALU"
D_Q.AND.RC[]      "RAMX/Q,AMX/RAMX.SPO.R_LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D&Q_D+Q           "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF,QK/SHF"
D_Q+D             "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF"
D_Q-D             "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_Q-D-1           "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B-1,SHF/ALU,DK/SHF"
D_Q[]D            "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/@1,SHF/ALU,DK/SHF"
D_Q(FRAC)         "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,DK/SHF.FL"
D_Q+K[]           "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF"
D_Q-K[]           "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
D_Q-K[]-1         "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1,SHF/ALU,DK/SHF"
D_Q+LB            "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B,SHF/ALU,DK/SHF"
D_Q[]MASK         "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/@1,SHF/ALU,DK/SHF"
D_Q.LEFT          "RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT,DK/SHF"
D_Q.OR.K[]        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF"
D_Q.ORNOT.MASK    "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,DK/SHF"
D_Q.OR.PSWC       "DK/SHF,ALU/OR,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.1,SHF/ALU"
```

```
D_Q.OR.RC[]       "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,DK/SHF"
D_Q+PC            "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,DK/SHF"
D_Q-PCSV          "RAMX/Q,AMX/RAMX,BMX/0,MSC/READ.RLOG,ALU/A-B,SHF/ALU,DK/SHF"
D_Q.RIGHT         "RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT,DK/SHF"
D_Q.RIGHT2        "RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT2,DK/SHF"
D_Q.SXT[]         "RAMX/Q,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_Q.XOR.RC[]      "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF"
;D_R... THRU D&VA_...

D_R[]             "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_R[].AND.K[]     "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_R[].ORNOT.K[]   "LAB_R[@1],AMX/LA,BMX/KMX,KMX/@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.LAB.SPO.RAB/@1,AMX/LA,ALU/A.SHF/ALU,DK/SHF.FL"
D_RLOG            "BMX/0,MSC/READ.RLOG,ALU/B,SHF/ALU,DK/SHF"
D_RLOG.RIGHT      "BMX/0.MSC/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"
D&VA_ALU          "VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_D-K[]        "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_D+LC         "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_D+Q          "D_D+Q,VAK/LOAD"
D&VA_LA           "AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_LB           "BMX/LB,ALU/B,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_Q            "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,DK/Q"
D&VA_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,MSC/LOAD.STATE"

FE_0(A)           "AMX/RAMX.0XT,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_NABS(SC-LA(EXP)) "AMX/LA,EBMX/AMX.EXP,EALU/NABS.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.KMX,ID.ADDR/@1"
ID_D&NO.SYNC      "CID/WRITE.KMX,ADS/IBA,KMX/SP1.CON"
ID_D.SYNC         "CID/WRITE.KMX,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)&LB_R(SRC)       "SPO.AC/LOAD.LAB,SPO.ACN11/DST.SRC"
LA_R(SP2)&LB_R(SP1)       "SPO.AC/LOAD.LAB,SPO.ACN/SP2.SP1"
LAB_R1&RC[]_0             "ALU_0(A),LAB_R1&RC[@1]_ALU"
LAB_R1&RC[]_ALU "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/@1,SHF/ALU"
```

```
LAB_R1&RC[]_ALU.RIGHT2  "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/@1,SHF/RIGHT2"
LAB_R1&RC[]_D.0XT[]+K[] "ALU_D.0XT[@2]+K[@3],LAB_R1&RC[@1]_ALU"
LAB_R1&RC[]_D+LC        "ALU_D+LC,LAB_R1&RC[@1]_ALU"
LAB_R1&RC[]_Q-K[]       "ALU_Q-K[@2],LAB_R1&RC[@1]_ALU"
LAB_R1&RC[]_0-D "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/@1,ALU/A-B,AMX/RAMX.0XT,DT/LONG,BMX/RBMX,RBMX/D,SHF/ALU"
LAB_R1&RC[]_0+LC+1      "ALU/A+B+1,AMX/RAMX.0XT,DT/LONG,BMX/LC,SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/@1,SHF/ALU"
LAB_R[]         "SPO.R/LOAD.LAB,SPO.RAB/@1"
LAB_R(DST)      "SPO.AC/LOAD.LAB,SPO.ACN11/DST.DST"
LAB_R(SC)       "SPO.AC/LOAD.LAB,SPO.ACN/SC"
LAB_R(SP1)      "SPO.AC/LOAD.LAB,SPO.ACN/SP1.SP1"
LC_RC[]         "SPO.R/LOAD.LC,SPO.RC/@1"
LC_RC[]&R1_D            "ALU_D,LC_RC[@1]&R1_ALU"
LC_RC[]&R1_LA-K[]       "ALU_LA-K[@2],LC_RC[@1]&R1_ALU"
LC_RC(SC)       "SPO/LOAD.LC.SC"
LC_RC[]&R1_ALU  "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU"
LC_RC[]&R1_(LA+LB).LEFT "AMX/LA,BMX/LB,ALU/A+B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
LC_RC[]&R1_(LA-LB).LEFT "AMX/LA,BMX/LB,ALU/A-B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
LC_RC[]&R1_LA+K[]       "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU,ALU/A+B,AMX/LA,BMX/KMX,KMX/@2"
LC_RC[]&R1_LB   "ALU_LB,LC_RC[@1]&R1_ALU"
LC_RC[]&R1_Q    "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU,ALU/A,AMX/RAMX,RAMX/Q"
;PC_... THRU PC&VA_...

PC_PC+1         "PCK/PC+1"
PC_PC+2         "PCK/PC+2"
PC_PC+4         "PCK/PC+4"
PC_PC+N         "PCK/PC+N"
PC_Q+PC         "ALU/A+B,VAK/LOAD,PCK/PC_VA,BMX/PC,AMX/RAMX,RAMX/Q"
PC_VA           "PCK/PC_VA"
PC&VA_ALU       "VAK/LOAD,PCK/PC_VA"
PC&VA_D         "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
PC&VA_D.0XT[]   "RAMX/D,AMX/RAMX.0XT,DT/@1,ALU/A,VAK/LOAD,PCK/PC_VA"
PC&VA_D.0XT[]+PC        "RAMX/D,AMX/RAMX.0XT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_D.SXT[]+PC "RAMX/D,AMX/RAMX.SXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_D+K[]     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_D-K[]     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
PC&VA_D-PC      "RAMX/D,AMX/RAMX,BMX/PC,ALU/A-B,VAK/LOAD,PCK/PC_VA"
PC&VA_K[]       "KMX/@1,BMX/KMX,ALU/B,VAK/LOAD,PCK/PC_VA"
PC&VA_PC        "BMX/PC,ALU/B,VAK/LOAD,PCK/PC_VA"
PC&VA_Q         "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
PC&VA_Q-D       "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
```

```
PC&VA_Q-K[]      "RAMX/Q,AMX/RAMX,KMX,@1,BMX/KMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
PC&VA_Q+PC       "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_Q.SXT[]+PC         "RAMX/Q,AMX/RAMX.SXT,DT @1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VA_R[].ANDNOT.K[] "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/ANDNOT,VAK/LOAD,PCK/PC_VA"
PC&VA_RC[]       "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,VAK/LOAD,PCK/PC_VA"
PC_VIBA          "PCK/PC_IBA"

;Q_0... THRU Q_D...

Q_0              "QK/CLR"
Q_0-D            "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-K[]          "AMX/RAMX.0XT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-LC           "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_0+MASK+1       "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,QK/SHF"
Q_0+PC.RLOG      "AMX/RAMX.0XT,DT/LONG,BMX/PC,ALU/A+B.RLOG,SHF/ALU,QK/SHF"
Q_0-Q            "AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_ACCEL&SYNC     "QK/ACCEL,ACF/SYNC"
Q_ALU            "SHF/ALU,QK/SHF"
Q_ALU.LEFT       "SHF/LEFT,QK/SHF"
Q_ALU.LEFT2      "SHF/ALU.DT,DT/LONG,QK/SHF"
Q_ALU.RIGHT      "SHF/RIGHT,QK/SHF"
Q_ALU.RIGHT2     "SHF/RIGHT2,QK/SHF"
Q_ALU(FRAC)      "SHF/ALU,QK/SHF.FL"
Q_D              "QK/D"
Q_D(FRAC)(B)     "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,QK/SHF.FL"
Q_D.0XT[]        "RAMX/D,AMX/RAMX.0XT,DT/@1,ALU/A,SHF/ALU,QK/SHF"
Q_D.0XT[]+K[].LEFT "RAMX/D,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B,SHF/LEFT,QK/SHF"
Q_D.AND.K[]      "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_D.AND.K[].RIGHT        "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,QK/SHF"
Q_D.AND.K[].RIGHT2       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT2,QK/SHF"
Q_D.ANDNOT.RC[] "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_D.AND.RC[]     "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,QK/SHF"
Q_DEC.CON        "QK/DEC.CON"
Q_D+K[]          "RAMX/D,AMX/RAMX,KMX/@1,BMX,KMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_D+K[].LEFT     "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/LEFT,QK/SHF"
Q_D-K[]          "RAMX/D,AMX/RAMX,KMX/@1,BMX,KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_D+LC           "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF"
Q_D-LC           "RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_D.LEFT3        "RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT3,QK/SHF"
Q_D.OR.K[]       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,QK/SHF"
```

```
Q_D.OR.RC[]      "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,QK/SHF"
Q_D-Q            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_D.RIGHT        "RAMX/D,AMX/RAMX,ALU/A,SHF/RIGHT,QK/SHF"
Q_D.RIGHT2       "RAMX/D,AMX/RAMX,ALU/A,SHF/RIGHT2,QK/SHF"
Q_D.SXT[]        "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,QK/SHF"
Q_D.XOR.Q        "QK/SHF,ALU/XOR,AMX/RAMX,RAMX/D,BMX/RBMX,RBMX/Q,SHF/ALU"
;Q_IB... THRU Q_PC...

Q_IB.BDEST       "IBC/BDEST,QK/ID,MCT/ALLOW.IB.READ"
Q_IB.DATA        "QK/ID,MCT/ALLOW.IB.READ"
Q_ID[]           "CID/READ.KMX,ID.ADDR/@1,QK/ID"
Q_ID(SC)         "CID/READ.SC,QK/ID"
Q_K[]            "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,QK/SHF"
Q_K[].CTX        "KMX/@1,BMX/KMX,ALU/B,SHF/ALU.DT,DT/INST.DEP,QK/SHF"
Q_K[].RIGHT      "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT,QK/SHF"
Q_K[].RIGHT2     "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT2,QK/SHF"
Q_LA             "AMX/LA,ALU/A,SHF/ALU,QK/SHF"
Q_LA.AND.K[]     "AMX/LA,KMX/@1,BMX/KMX,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+K[]         "AMX/LA,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_LA-K[]         "AMX/LA,KMX/@1,BMX/KMX,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_RA[@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 Q&VA_...

Q_Q.0XT[]-K[]    "RAMX/Q,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q.0XT[].LEFT   "RAMX/Q,AMX/RAMX.0XT,DT/@1,ALU/A,SHF/LEFT,QK/SHF"
Q_Q.0XT[].OR.D   "RAMX/Q,AMX/RAMX.0XT,DT/@1,RBMX/D,BMX/RBMX,ALU/OR,SHF/ALU,QK/SHF"
Q_Q.AND.K[]      "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_Q.AND.K[].RIGHT        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,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.K[]   "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,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+K[]          "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q-K[]          "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q-K[]-1        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q+LC           "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF"
Q_Q-LC           "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q-LC-1         "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q.LEFT         "QK/LEFT"
Q_Q-MASK-1       "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU,QK/SHF"
Q_Q.OR.K[]       "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,QK/SHF"
Q_Q.ORNOT.MASK   "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,QK/SHF"
Q_Q+PC           "RAMX/Q,AMX/RAMX,BMX/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,AMX/LA,ALU/A,SHF/ALU,QK/SHF"
Q_R[].AND.K[]    "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/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,BMX/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,AMX/LA,ALU/A,SHF/ALU,QK/SHF.FL"
Q_R(PRN).ANDNOT.Q        "SPO.AC/LOAD.LAB,SPO.ACN/PRN,AMX/LA,RBMX/Q,BMX/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(SRC!1).AND.K[]       "SPO.AC/LOAD.LAB,SPO.ACN11/SRC.CR.1,AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF"
Q_SC             "ALU/B,BMX/KMX,KMX/SC,SHF/ALU,QK/SHF"
Q&VA_ALU         "VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_D           "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_D+LC        "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_LA          "AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_Q+LB.PC     "RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF"
;R[]_0...THRU R[]_PACK.FP

R[]_0            "SPO.R/WRITE.RAB,SPO.RAB/@1,AMX/RAMX.SXT,DT/LONG,ALU/A,SHF/ALU"
R[]_0-D          "AMX/RAMX.SXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-K[]        "AMX/RAMX.SXT,DT/LONG,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-LB         "AMX/RAMX.SXT,DT/LONG,BMX/LB,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
```

```
R[]_0+LB+1        "AMX/RAMX.OXT,DT/LONG,BMX/LB,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-Q           "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-1           "AMX/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"
R6_D+K[].RLOG     "SPO.R/WRITE.RAB,SPO.RAB/R6,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"
R6_LA+K[].RLOG    "AMX/LA,BMX/KMX,KMX/@1,ALU/A+B.RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/R6"
R6_LA-K[].RLOG    "AMX/LA,BMX/KMX,KMX/@1,ALU/A-B.RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/R6"
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.ORNOT.MASK         "AMX/LA,BMX/MASK,ALU/ORNOT,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.0         "AMX/RAMX.0XT,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.0XT,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,RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU"
R[]_Q+1           "ALU_Q+Q+1,R[@1]_ALU"
R[]_Q+5           "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/A+B+1,BMX/KMX,KMX/.4,AMX/RAMX,RAMX/Q,SHF/ALU"
R[]_Q-D           "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU"
R[]_Q-D-1         "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D,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[]         "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,AMX/RAMX,BMX/KMX,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,RAMX/Q,SHF/ALU"
R[]_Q+LC          "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU"
R[]_Q-LC          "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU"
R[]_Q.AND.K[]     "ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/@1,AMX/RAMX,RAMX/Q,BMX/RBMX,KMX/@2"
R[]_Q.ANDNOT.K[]        "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@2,SHF/ALU"
R[]_Q.OR.D        "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/OR,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D,SHF/ALU"
R[]_Q.ORNOT.K[]   "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/KMX,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        "BMX/0,MSC/READ.RLOG,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1"
;RC[]_0... THRU RC[]_D...

RC[]_0            "AMX/RAMX.0XT,DT/LONG,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0-D          "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0+K[]+1      "AMX/RAMX.0XT,DT/LONG,KMX/@2,BMX/KMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0+LC+1       "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0+MASK+1     "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_0+MASK+1.RIGHT2      "AMX/RAMX.0XT,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.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            "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.0XT[]      "RAMX/D,AMX/RAMX.0XT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.AND.K[]    "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
```

```
RC[]_D.AND.MASK "RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.ANDNOT.Q "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D(B)       "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.CTX      "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D+K[]      "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D-K[]      "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.LEFT     "RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.LEFT3    "RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.OR.K[]   "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.OR.Q     "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_D.ORNOT.K[]        "SPO.RC/@1,SPO.R/WRITE.RC,ALU/ORNOT,AMX/RAMX,RAMX/D,BMX/KMX,KMX/@2,SHF/ALU"
RC[]_D.SXT[]    "RAMX/D,AMX/RAMX.SXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;RC[]_K... THRU RC[]&VA_...

RC[]_K[]        "KMX/@2,BMX/KMX,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_K[]+1      "AMX/RAMX.OXT,DT/LONG,KMX/@2,BMX/KMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_K[].LEFT2  "KMX/@2,BMX/KMX,ALU/B,SHF/ALU.DT,DT/LONG,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_K[].RIGHT2 "KMX/@2,BMX/KMX,ALU/B,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LA         "AMX/LA,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LA.AND.K[]         "ALU_LA.AND.K[@2],RC[@1]_ALU"
RC[]_LA.CTX     "AMX/LA,ALU/A,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LA-K[]     "AMX/LA,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LB         "BMX/LB,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LB.LEFT    "BMX/LB,ALU/B,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LC         "BMX/LC,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_NOT.Q      "RAMX/Q,AMX/RAMX,ALU/NOTA,RC[@1]_ALU"
RC[]_PACK.FP    "BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_PC         "BMX/PC,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q          "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.OXT[]    "RAMX/Q,AMX/RAMX.OXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.AND.K[]  "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.ANDNOT.K[]       "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q+1        "ALU_Q+Q+1,RC[@1]_ALU"
RC[]_Q+K[]      "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q-K[]      "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.LEFT     "RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.LEFT3    "RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q-MASK-1   "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q+PC       "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q+PC+1     "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
```

```
RC[]_Q.RIGHT      "RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_Q.SXT[]      "RAMX/Q,AMX/RAMX.SXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_RLOG.RIGHT "BMX/0,MSC/READ.RLOG,ALU/B,SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]&VA_D+Q       "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC(SC)_ALU        "SHF/ALU,SPO/WRITE.RC.SC"
;R(DST)_... THRU R[]&VA_...
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.SXT[].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,BMX/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_LC,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"
R(SRC)_LC         "BMX/LC,ALU/B,R(SRC)_ALU"
R(SRC)_Q          "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
R(SRC!1)_ALU      "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.OR.1"
R(SRC!1)_D(B)     "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.OR.1"
R[]&VA_LA+K[]     "AMX/LA,KMX/@2,BMX/KMX,ALU/A+B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]&VA_LA-K[]     "AMX/LA,KMX/@2,BMX/KMX,ALU/A-B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]&VA_LA-K[].RLOG      "AMX/LA,KMX/@2,BMX/KMX,ALU/A-B.RLOG,DT/LONG,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]&VA_Q-K[]      "RAMX/Q,AMX/RAMX,KMX/@2,BMX/KMX,ALU/A-B,VAK/LOAD,SPO.R/WRITE.RAB,SPO.RAB/@1"
;SC_...

SC_0(A)           "AMX/RAMX.0XT,DT/LONG,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
SC_0-K[]          "BMX/KMX,KMX/@1,AMX/RAMX.0XT,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              "RAMX/D,AMX/RAMX,ALU/A,SMX/ALU,SCK/LOAD"
SC_D.0XT[]-K[]    "RAMX/D,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_D.0XT[].XOR.K[]      "RAMX/D,AMX/RAMX.0XT,DT/@1,BMX/KMX,KMX/@2,ALU/XOR,SC_ALU"
SC_D.AND.K[]      "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD"
SC_D(EXP)         "RAMX/D,AMX/RAMX,ALU/A,SMX/ALU.EXP,SCK/LOAD"
SC_D(EXP)(A)      "RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
SC_D(EXP)(B)      "RBMX/D,BMX/RBMX,ALU/B,SMX/ALU.EXP,SCK/LOAD"
SC_D-K[]          "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_D.OR.K[]       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SMX/ALU,SCK/LOAD"
SC_D.SXT[]        "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SMX/ALU,SCK/LOAD"
SC_EALU           "SMX/EALU,SCK/LOAD"
SC_FE             "SMX/FE,SCK/LOAD"
SC_NABS(SC-FE)    "EBMX/FE,EALU/NABS.A-B,SMX/EALU,SCK/LOAD"
SC_K[]            "KMX/@1,EBMX/KMX,EALU/B,SMX/EALU,SCK/LOAD"
SC_K[].ALU        "KMX/@1,BMX/KMX,ALU/B,SMX/ALU,SCK/LOAD"
SC_LA             "AMX/LA,ALU/A,SMX/ALU,SCK/LOAD"
SC_LA.AND.K[]     "AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD"
SC_LC(EXP)        "BMX/LC,ALU/B,SMX/ALU.EXP,SCK/LOAD"
SC_PSLADDR        "SMX/EALU,EBMX/KMX,SCK/LOAD,KMX/.F,EALU/B"
SC_Q              "RAMX/Q,AMX/RAMX,ALU/A,SMX/ALU,SCK/LOAD"
SC_Q.AND.K[]      "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/AND,SMX/ALU,SCK/LOAD"
SC_Q(EXP)         "RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
```

```
SC_Q(EXP)(B)      "RBMX/Q,BMX/RBMX,ALU/B,SMX/ALU.EXP,SCK/LOAD"
SC_Q+K[]          "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/A+B,SMX/ALU,SCK/LOAD"
SC_Q-K[]          "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/A-B,SMX/ALU,SCK/LOAD"
SC_Q.OR.K[]       "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/OR,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.K[]    "ALU/AND,AMX/LA,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/KMX,KMX/@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) "LAB_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_0&SD_0         "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.0XT,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.VIA.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  "EBMX/FE,EALU/A+B,MSC/LOAD.STATE"
STATE_STATE-FE  "EBMX/FE,EALU/A-B,MSC/LOAD.STATE"
STATE_STATE+K[] "KMX/@1,EBMX/KMX,EALU/A+B,MSC/LOAD.STATE"
STATE_STATE-K[] "KMX/@1,EBMX/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_SKPLONG           "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.STOO     "STATE_STATE.ANDNOT.K[.3F]"
STATE_STATE.AN.6TO4     "STATE_STATE.ANDNOT.K[.7F]"
STATE_STATE.AN.DESTDBL  "STATE_STATE.ANDNOT.K[.6]"
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[.3]"
STATE_STATE.OR.DEST     "STATE_STATE.OR.K[.4]"
STATE_STATE.OR.DESTDBL  "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.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_INNEROBJ  "STATE_K[.1]"
STATE_INNERSRC  "STATE_K[.3]"
STATE_OUTER     "STATE_K[ZERO]"

SWAPD           "DK/BYTE.SWAP"

VA_ALU          "VAK/LOAD"
VA_D            "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD"
VA_D.OXT[]+Q    "RAMX/D,AMX/RAMX,OXT,DT/@1,BMX/RBMX,ALU/A+B,VAK/LOAD"
VA_D.ANDNOT.K[] "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@1,ALU/ANDNOT,VAK/LOAD"
VA_D+K[]        "RAMX/D,AMX/RAMX,KMX/@1,BMX,KMX,ALU/A+B,VAK/LOAD"
VA_D+LC         "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD"
VA_D+Q          "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD"
```

```
VA_K[]              "KMX/@1,BMX/KMX,ALU/B,VAK/LOAD"
VA_LA               "AMX/LA,ALU/A,VAK/LOAD"
VA_LA.AND.LC        "AMX/LA,BMX/LC,ALU/AND,VAK/LOAD"
VA_LA.ANDNOT.K[]            "AMX/LA,BMX/KMX,KMX/@1,ALU/ANDNOT,VAK/LOAD"
VA_LA+D             "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,VAK/LOAD"
VA_LA-D             "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,VAK/LOAD"
VA_LA+K[]           "AMX/LA,BMX/KMX,KMX/@1,ALU/A+B,VAK/LOAD"
VA_LA-K[]           "AMX/LA,BMX/KMX,KMX/@1,ALU/A-B,VAK/LOAD"
VA_LA+K[]+1         "AMX/LA,BMX/KMX,KMX/@1,ALU/A+B+1,VAK/LOAD"
VA_LA-K[]-1         "AMX/LA,BMX/KMX,KMX/@1,ALU/A-B-1,VAK/LOAD"
VA_LA+PC            "AMX/LA,BMX/PC,ALU/A+B,VAK/LOAD"
VA_LA+Q             "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD"
VA_LA-Q             "VAK/LOAD,ALU/A-B,AMX/LA,BMX/RBMX,RBMX/Q,SHF/ALU"
VA_LB+D.OXT         "BMX/LB,ALU/A+B,AMX/RAMX.OXT,DT/BYTE,VAK/LOAD"
VA_PC               "BMX/PC,ALU/B,VAK/LOAD"
VA_Q                "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD"
VA_Q.ANDNOT.K[]     "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,VAK/LOAD"
VA_Q+D              "VAK/LOAD,ALU/A+B,AMX/RAMX,BMX/RBMX,RAMX/Q,RBMX/D,SHF/ALU"
VA_Q+K[]            "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD"
VA_Q-K[]            "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD"
VA_Q+LC             "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD"
VA_Q+LB             "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B,VAK/LOAD"
VA_Q-LB             "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A-B,VAK/LOAD"
VA_Q+LB.PC          "RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD"
VA_Q+PC             "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD"
VA_R[]              "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,VAK/LOAD"
VA_RC[]             "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,VAK/LOAD"
VA_VA+4             "PCK/VA+4"
        "Non-transfer Functions"

B.FORK              "LAB_R(SP1),QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/B.FORK"
BYTE                "DT/BYTE"
CACHE.INVALIDATE            "MCT/INVALIDATE,VAK/NOP"
CALL                "SUB/CALL"
CALL[]              "CALL,J/@1"
C.FORK              "SUB/SPEC,J/C.FORK"
CHK.ODD.ADDR        "MSC/CHK.ODD.ADDR"
CHK.FLT.OPR         "MSC/CHK.FLT.OPR"
CLK.UBCC            "CCK/LOAD.UBCC"
CLR.FPD             "MSC/CLR.FPD"
CLR.IB0-1           "IBC/CLR.0.1,IEK/ISTR"
```

```
CLR.IB0-3       "IBC/CLR.0-3"               ;DISCARD -11 INSTR & OPERAND
CLR.IB2-3       "IBC/CLR.2.3"               ;11 MODE DISCARD ISTREAM OPERAND
CLR.IB2-5       "IBC/CLR.1-5.COND"          ;2ND PART OF Q/D IMMEDIATE
CLR.IB.COND     "IBC/CLR.1-5.COND"
CLR.IB.OPC      "IBC/CLR.0,IEK/ISTR"
CLR.IB.SPEC     "IBC/CLR.1"
CLR.NEST.ERR    "MSC/CLR.NEST.ERR"
CLR.SD&SS       "SGN/CLR.SD+SS"
EXCEPT.ACK      "IEK/EACK"
FLUSH.IB        "IBC/FLUSH,VAK/LOAD,IEK/ISTR"
INHIBIT.IB      "MCT/MEM.NOP"
INTRPT.ACK      "IEK/IACK"
INTRPT.STROBE   "IEK/ISTR"
IRD             "IRD0,CLR.UBCC,IRD1,SUB/SPEC,J/A.FORK"
IRD0            "LA_R(SP2)&LB_R(SP1),D&VA_LB,SC_ALU(EXP),FE_LA(EXP),SS_ALU15"
IRD1            "MSC/IRD,QK/ID,MCT/ALLOW.IB.READ,IBC/CLR.1-5.COND,PCK/PC+N"
IRD.11          "LA_R(DST)&LB_R(SRC),D_LB.PC,VAK/LOAD,Q_IB.DATA,SC_K[.10],PCK/PC+N,MSC/IRD,SUB/SPEC,J/DP0"
LOAD.IB         "VAK/NOP,MCT/READ.V.NEWPC"
LOAD.IB.11      "VAK/NOP,MCT/READ.V.NEWPC"
LONG            "DT/LONG"
POLY.DONE       "ACF/CONTROL,ACM/POLY.DONE"
RETURN[]        "SUB/RET,J/@1"
RETURN0         "SUB/RET,J/0"
RETURN1         "SUB/RET,J/1"
RETURN2         "SUB/RET,J/2"
RETURN3         "SUB/RET,J/3"
RETURN8         "SUB/RET,J/8"
RETURN9         "SUB/RET,J/9"
RETURNF         "SUB/RET,J/0F"
RETURN10        "SUB/RET,J/10"
RETURN12        "SUB/RET,J/12"
RETURN18        "SUB/RET,J/18"
RETURN1F        "SUB/RET,J/1F"
RETURN20        "SUB/RET,J/20"
RETURN24        "SUB/RET,J/24"
RETURN40        "SUB/RET,J/40"
RETURN60        "SUB/RET,J/60"
RETURN61        "SUB/RET,J/61"
RETURN100       "SUB/RET,J/100"
RETURN10C       "SUB/RET,J/10C"
```

```
RETURN10E       "SUB/RET,J/10E"
SET.CC(INST)    "CCK/INST.DEP,DT/INST.DEP"
SET.CC(LONG)    "CCK/INST.DEP,DT/LONG"
SET.CC(ROR)     "CCK/ROR"
SET.FPD         "MSC/SET.FPD"
SET.N.AND.Z     "CCK/TST.Z"
SET.NEST.ERR    "MSC/SET.NEST.ERR"
SET.N&Z         "CCK/N+Z_ALU"
SET.PSL.C(AMX)  "CCK/C_AMX0"
SET.V           "CCK/SET.V"
START.IB        "IBC/START"
STOP.IB         "IBC/STOP"
TEST.TB.RCHK    "MCT/TEST.RCHK,VAK/NOP"
TEST.TB.WCHK    "MCT/TEST.WCHK,VAK/NOP"
TRAP.ACC[]      "ACF/TRAP.ACM/@1"
WORD            "DT/WORD"
WRITE.DEST      "LAB_R(SP1),QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/WRD"
      "Branch Enable Macro Definitions"

ACCEL?          "BEN/ACCEL"
ACC.SYNC?       "BEN/ACCEL"       ;,J3/3"
AC.LOW?         "BEN/INTERRUPT"   ;,J3/3"
ALIGNED?        "BEN/TB.TEST"     ;,J5/17"
ALU?            "BEN/ALU"
ALU.N?          "BEN/ALU"         ;,J4/07"
ALU1-0?         "BEN/ALU1-0"
BCDSGN?         "BEN/DECIMAL"     ;,J2/2"
C31?            "BEN/C31"
CONSOLE.MODE?   "BEN/PSL.MODE"    ;,J5/18"
D0?             "BEN/D3-0"        ;,J4/0E"
D(1)?           "BEN/MUL"
D2?             "BEN/D3-0"        ;,J4/0B"
D2-0?           "BEN/D3-0"        ;,J4/08"
D3?             "BEN/D3-0"        ;,J4/07"
D3-0?           "BEN/D3-0"
D31?            "BEN/SIGNS"       ;,J3/6"
DATA.TYPE?      "BEN/DATA.TYPE"
D.B0?           "BEN/D.BYTES"     ;,J4/0E"
D.B1?           "BEN/D.BYTES"     ;,J4/0D"
D.B2?           "BEN/D.BYTES"     ;,J4/0B"
```

```
DBL?              "BEN/DATA.TYPE"
D.BYTES?          "BEN/D.BYTES"
D.NE.0?           "BEN/SIGNS"       ;,J3/5" ;PREFERED FORM
EALU?             "BEN/EALU"
EALU.N?           "BEN/EALU"        ;,J4/07"
EALU.Z?           "BEN/EALU"        ;,J4/0B"
END.DP1?          "BEN/END.DP1"
FPD?              "BEN/LAST.REF"    ;,J4/C7"
IB.TEST?          "BEN/IB.TEST"
INT?              "BEN/INTERRUPT"
INTERRUPT.REQ?    "BEN/INTERRUPT"   ;,J3/5"
IR0?              "BEN/ALU"         ;,J4/0D"
IR0.C31?          "BEN/ALU"
IR1?              "BEN/IR2-1"       ;,J3/6"
IR2-1?            "BEN/IR2-1"
LAST.REF?         "BEN/LAST.REF"
MODE.LSS.ASTLVL?          "BEN/REI"         ;,J3/3"
MUL?              "BEN/MUL"
NEST.ERR?         "BEN/LAST.REF"    ;,J4/0B"
PC.MODES?         "BEN/PC.MODES"
PSL.C?            "BEN/PSL.CC"      ;,J4/0E"
PSL.CC?           "BEN/PSL.CC"
PSL.MODE?         "BEN/PSL.MODE"
PSL.N?            "BEN/PSL.CC"      ;,J4/7"
PSL.V?            "BEN/PSL.CC"      ;,J4/0D"
PSL.Z?            "BEN/PSL.CC"      ;,J4/0B"
PTE.VALID?        "BEN/TB.TEST"     ;,J5/0F"
QUAD?             "BEN/DATA.TYPE"
Q31?              "BEN/SIGNS"       ;,J3/3"
RLOG.EMPTY?       "BEN/ALU1-0"      ;,J4/7"
ROR?              "BEN/ROR"
SC?               "BEN/SC"
SC.GT.0?          "BEN/SC"
SC.NE.0?          "BEN/MUL"         ;,J3/3"
SIGNS?            "BEN/SIGNS"
SRC.PC?           "BEN/SRC.PC"      ;COMP MODE, BEN ON SRC R = PC
SS?               "BEN/EALU"        ;,J4/0E"
STATE0?           "BEN/STATE3-0"    ;,J4/0E"
STATE1?           "BEN/STATE3-0"    ;,J4/0D"
STATE1-0?         "BEN/STATE3-0"    ;,J4/0C"
STATE2?           "BEN/STATE3-0"    ;,J4/0B"
```

```
STATE3?          "BEN/STATE3-0"   ;,04/07"
STATE3-0?        "BEN/STATE3-0"
STATE4?          "BEN/STATE7-4"
STATE5?          "BEN/STATE7-4"
STATE6?          "BEN/STATE7-4"
STATE(7)?              "STATE7-4?"
STATE7-4?        "BEN/STATE7-4"
TB.TEST?         "BEN/TB.TEST"
VA31?            "BEN/PSL.MODE"   ;,05/0F"
VA31-30?         "BEN/PSL.MODE"   ;,05/07"
Z?               "BEN/Z"
ZONED?           "BEN/DECIMAL"    ;,02/1"
ALEG_D           "AMX/RAMX,RAMX/D"
ALEG_LA          "AMX/LA"
ALEG_LAB         "AMX/LA"
ALEG_Q           "AMX/RAMX,RAMX/Q"
ALU_0(K)         "KMX/ZERO,BMX/KMX,ALU/B"
ALU_0+D+1        "AMX/RAMX.0XT.DT/LONG,RBMX/D,BMX/RBMX,ALU/A+B+1"
ALU_D.AND.Q      "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/AND"
ALU_D.OR.MASK    "RAMX/D,AMX/RAMX,BMX/MASK,ALU/OR"
ALU_D[]LB        "RAMX/D,AMX/RAMX,BMX/LB,ALU/@1"
ALU_D[]MASK      "RAMX/D,AMX/RAMX,BMX/MASK,ALU/@1"
ALU_D[]RC[]      "AMX/RAMX,RAMX/D,LC_RC[@2],BMX/LC,ALU/@1"
ALU_D[]R[]       "AMX/RAMX,RAMX/D,LAB_R[@2],BMX/LB,ALU/@1"
ALU_D-R[]        "BMX/RBMX,RBMX/D,LAB_R[@1],AMX/LA,ALU/A-B"
ALU_LA-LC        "AMX/LA,BMX/LC,ALU/A-B"
ALU_LA.AND.LC    "AMX/LA,BMX/LC,ALU/AND"
ALU_LA[]K[]      "AMX/LA,BMX/KMX,KMX/@2,ALU/@1"
ALU_LA[]LC       "AMX/LA,BMX/LC,ALU/@1"
ALU_MASK         "BMX/MASK,ALU/B"
ALU_MASK+1       "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1"
ALU_NOT.K[]      "AMX/RAMX.0XT,DT/LONG,KMX/@1,BMX/KMX,ALU/ORNOT"
ALU_NOT.LA       "AMX/LA,ALU/NOTA"
ALU_NOT.MASK     "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/ORNOT"
ALU_NOT.Q        "RAMX/Q,AMX/RAMX,ALU/NOTA"
ALU_Q+D "RAMX/Q,AMX/RAMX,BMX/RBMX,ALU/A+B"
ALU_Q-SC         "KMX/SC,BMX/KMX,RAMX/Q,AMX/RAMX,ALU/A-B"
ALU_Q[]LB        "RAMX/Q,AMX/RAMX,BMX/LB,ALU/@1"
ALU_Q[]MASK      "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/@1"
ALU_Q.AND.MASK   "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/AND"
```

```
ALU_Q.OR.MASK    "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/OR"
ALU_Q.ORNOT.MASK          "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT"
ALU_Q.XOR.RC[]   "LC_RC[@1],AMX/RAMX,RAMX/Q,BMX/LC,ALU/XOR"
ALU_Q.XOR.R(SC)  "LAB_R(SC),BMX/LB,AMX/RAMX,RAMX/Q,ALU/XOR"
ALU_Q[]K[]       "RAMX/Q,AMX/RAMX,KMX/@2,BMX/KMX,ALU/@1"
ALU_Q[]LC        "RAMX/Q,AMX/RAMX,BMX/LC,ALU/@1"
ALU_Q[]RC[]      "AMX/RAMX,RAMX/Q,LC_RC[@2],BMX/LC,ALU/@1"
ALU_Q[]R[]       "AMX/RAMX,RAMX/Q,LAB_R[@2],BMX/LB,ALU/@1"
ALU_R(SC)        "SPO.AC/LOAD.LAB,SPO.ACN/SC,AMX/LA,ALU/A"
ALU_RLOG         "MSC/READ.RLOG,ALU/B"
ALU_R[].AND.MASK          "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/MASK,ALU/AND"
ALU_R[].OR.MASK  "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/MASK,ALU/OR"
ALU_R[].ORNOT.MASK        "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/MASK,ALU/ORNOT"
ALU_R[]-K[]      "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/A-B"
ALU_R[]+K[]      "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/A+B"
ALU_SC           "KMX/SC,BMX/KMX,ALU/B"
BLEG_D           "BMX/RBMX,RBMX/D"
BLEG_LB          "BMX/LB"
BLEG_LC          "BMX/LC"
BLEG_Q           "BMX/RBMX,RBMX/Q"
CPSYNC           "ACF/SYNC"
DELAY            "CALL[DELAY]"
D_0-K[]          "AMX/RAMX.0XT,KMX/@1,BMX/KMX,ALU/A-B,D_ALU"
D_ACC            "DK/ACCEL"
D_D-LB           "RAMX/D,AMX/RAMX,BMX/LB,ALU/A-B,D_ALU"
D_D[]RC[]        "ALU_D[@1]RC[@2],D_ALU"
D_D.AND.R[]      "RBMX/D,BMX/RBMX,SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/AND,D_ALU"
D_D.LEFT.Q       "DK/LEFT,SI/DIV"
D_D.OR.R[]       "RBMX/D,BMX/RBMX,SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/OR,D_ALU"
D_D.SXT[]+K[]    "RAMX/D,AMX/RAMX,SXT,DT,@1,KMX/@2,BMX/KMX,ALU/A+B,D_ALU"
D_D.XOR.RC[]     "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR,D_ALU"
D_LA+D           "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,D_ALU"
D_LA.OR.D        "AMX/LA,RBMX/D,BMX/RBMX,ALU/OR,D_ALU"
D_MASK           "BMX/MASK,ALU/B,D_ALU"
D_MASK+1         "AMX/RAMX.0XT,BMX/MASK,ALU/A+B+1,D_ALU"
D_NOT.RC[]       "AMX/RAMX.0XT,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ORNOT,D_ALU"
D_Q.OR.D         "RAMX/Q,AMX/RAMX,ALU/OR,SHF/ALU,DK/SHF,BMX/RBMX,RBMX/D"
D_MD             "DK/NOP"
D_R[].LEFT       "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/LEFT,DK/SHF"
D_R[].OR.D       "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,RBMX/D,BMX/RBMX,ALU/OR,D_ALU"
```

```
D_R[].ORNOT.MASK "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/MASK,ALU/ORNOT,D_ALU"
D_SC     "KMX/SC,BMX/KMX,ALU/B,SHF/ALU,DK/SHF"
EALU_STATE-K[]   "EBMX/KMX,KMX/@1,MSC/LOAD.STATE,EALU/A-B"
EBMX_K[]         "KMX/@1,EBMX/KMX"
ENDOVR   "SUB/CALL,J/SCOPE"
ERLOOP   "SUB/CALL,J/ERLOOP"
ERROR1   "SUB/CALL,J/ERROR1"
ERROR2   "SUB/CALL,J/ERROR2"
LAB_R(PRN)       "SPO.AC/LOAD.LAB,SPO.ACN/PRN"
LAB_R(PRN+1)     "SPO.AC/LOAD.LAB,SPO.ACN/PRN+1"
LAB_R(SP1+1)     "SPO.AC/LOAD.LAB,SPO.ACN/SP1+1"
LAB_R(SRC!1)     "SPO.AC/LOAD.LAB,SPO.ACN11/SRC.OR.1"
LAB_R(SP2)       "SPO.AC/LOAD.LAB,SPO.ACN/SP2.SP2"
LAB_R(SP2.SP1)   "SPO.AC/LOAD.LAB,SPO.ACN/SP2.SP1"
MESSAGE1         "D_K[ZERO],CALL[MSGCOM]"
MESSAGE2         "D_K[.1],CALL[MSGCOM]"
MESSAGE3         "D_K[.2],CALL[MSGCOM]"
MESSAGE4         "D_K[.3],CALL[MSGCOM]"
MESSAGE5         "D_K[.4],CALL[MSGCOM]"
MESSAGE7         "D_K[.6],CALL[MSGCOM]"
MESSAGE8         "D_K[.7],CALL[MSGCOM]"
MESSAGE9         "D_K[.8],CALL[MSGCOM]"
MESSAGE10        "D_K[.9],CALL[MSGCOM]"
NEWTST   "SUB/CALL,J/SCOPE"
NEWTST[]         "NEWTST,RC[OF]_K[@1]"
NOP      "DK/NOP"
Q_ACC            "QK/ACCEL"
Q_D[]Q           "RAMX/D,AMX/RAMX,BMX/RBMX,RBMX/Q,ALU/@1,Q_ALU"
Q_D[]RC[]        "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@2,BMX/LC,ALU/@1,Q_ALU"
Q_K(SP1)         "KMX/SP1.CON,BMX/KMX,ALU/B,Q_ALU"
Q_MASK+1         "AMX/RAMX.OXT,BMX/MASK,ALU/A+B+1,Q_ALU"
q_not.mask       "amx/ramx.0xt,bmx/mask,alu/ornot,q_alu"
Q_Q.AND.LC       "RAMX/Q,AMX/RAMX,BMX/LC,ALU/AND,Q_ALU"
Q_Q.AND.MASK     "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/AND,Q_ALU"
Q_Q.AND.R[]      "RBMX/Q,BMX/RBMX,SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/AND,Q_ALU"
Q_Q.ANDNOT.RC[]  "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,Q_ALU"
Q_Q.LEFT2        "QK/LEFT2"
Q_Q.XOR.D        "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/XOR,D_ALU"
Q_Q.XOR.R[]      "QK/SHF,AMX/RAMX,RAMX/Q,BMX/LB,ALU/XOR,LAB_R[@1]"
RC[]_0-K[]       "AMX/RAMX.OXT,KMX/@2,BMX/KMX,ALU/A-B,RC[@1]_ALU"
```

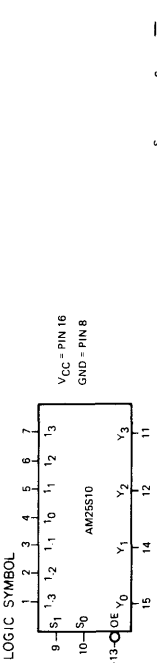
```
RC[]_D+Q          "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,RC[@1]_ALU"
RC[]_D.AND.LA     "RBMX/D,BMX/RBMX,AMX/LA,ALU/AND,RC[@1]_ALU"
RC[]_D.ORNOT.MASK "ALU_D.ORNOT.MASK,RC[@1]_ALU"
RC[]_D.XOR.LAB    "ALEG_D,BLEG_LB,ALU/XOR,RC[@1]_ALU"
RC[]_D.XOR.LC     "ALEG_D,BLEG_LC,ALU/XOR,RC[@1]_ALU"
RC[]_D+MASK+1     "RAMX/D,AMX/RAMX,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_LA+D         "AMX/LA,RBMX/D,BMX/RBMX,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.0        "AMX/RAMX.OXT,DT/LONG,ALU/NOTA,RC[@1]_ALU"
RC[]_NOT.D        "ALU_NOT.D,RC[@1]_ALU"
RC[]_NOT.MASK     "BMX/MASK,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT,RC[@1]_ALU"
RC[]_Q+LC         "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,RC[@1]_ALU"
RC[]_Q.AND.LA     "RBMX/Q,BMX/RBMX,AMX/LA,ALU/AND,RC[@1]_ALU"
RC[]_Q.OR.SC      "RAMX/Q,AMX/RAMX,KMX/SC,BMX/KMX,ALU/OR,RC[@1]_ALU"
RC[]_Q+MASK       "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A+B,RC[@1]_ALU"
RC[]_Q+MASK+1     "RAMX/Q,AMX/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       "RAMX/D,AMX/RAMX,BMX/RBMX,ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_D.AND.RC[]    "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@2,BMX/LC,ALU/AND,R[@1]_ALU"
R[]_D.ANDNOT.MASK "RAMX/D,AMX/RAMX,BMX/MASK,ALU/ANDNOT,R[@1]_ALU"
R[]_D.ANDNOT.Q    "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,R[@1]_ALU"
R[]_D.LEFT        "RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT,SPO.RAB/@1,SPO.R/WRITE.RAB"
R[]_D.OR.K[]      "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/OR,R[@1]_ALU"
R[]_D.ORNOT.MASK  "RAMX/D,AMX/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.K[]     "R[@1]_ALU,AMX/LA,BMX/KMX,KMX/@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"
RETURN4           "SUB/RET,J/4"
SC_SC-1           "KMX/.1,EBMX/KMX,EALU/A-B,SMX/EALU,SCK/LOAD"
SETMCR[]          "R[0A]_K[@1],CALL,J/SETMCR"
```

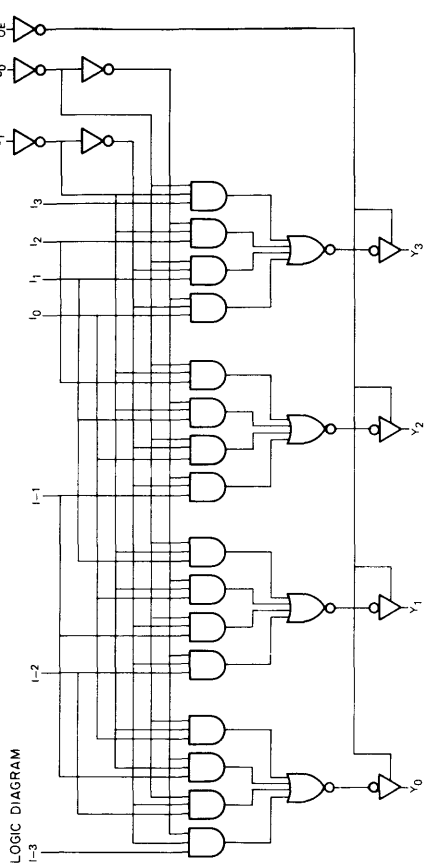
```
        SETRCF[]        "SC_K[@1],CALL,J/SETRCF"
        SETRXCS[]       "R[0A]_K[@1],CALL,J/SETRXCS"
        SETTXCS[]       "R[0A]_K[@1],CALL,J/SETTXCS"
        SUBTEST "CALL,J/SUBTST"
        TRAP.FPA        "ACF/CONTROL,ACM/7"
        VA_D.0XT[]      "ALU_D.0XT[@1],VA_ALU"
        VA_D.0XT[]+K[]  "ALU_D.0XT[@1]+K[@2],VA_ALU"
        VA_R[]+K[]      "SPO.R/LOAD.LAB,SPO.RA@/@1,KMX/@2,BMX/KMX,AMX/LA,ALU/A+B,VA_ALU"

; BRANCH DEFINITIONS
        ALU.Z?          "BEN/ALU"       ; NOTE J/XXB NESSECARY
        ALU.C?          "BEN/ALU"
        D1-0?           "BEN/D3-0"
```

CHAPTER 9

MISCELLANEOUS

25S10 FOUR-BIT SHIFTER CHIP WITH TRISTATE OUTPUT





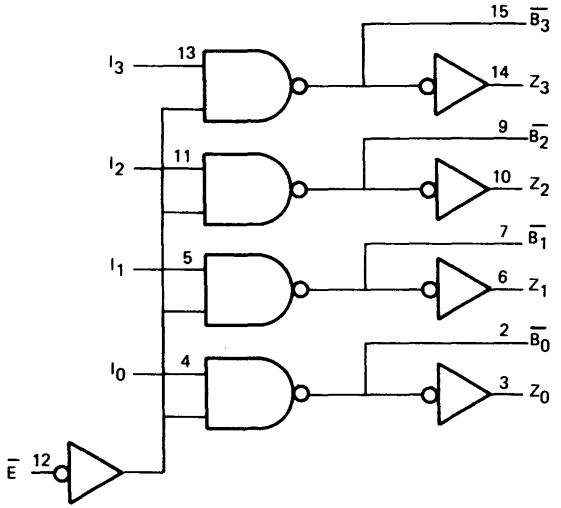
TRUTH TABLE

$\overline{OE}$	S_1	S_0	I_3	I_2	I_1	I_0	I_{-1}	I_{-2}	I_{-3}	Y_3	Y_2	Y_1	Y_0
H	X	X	X	X	X	X	X	X	X	Z	Z	Z	Z
L	L	L	D_3	D_2	D_1	D_0	X	X	X	D_3	D_2	D_1	D_0
L	L	H	X	D_2	D_1	D_0	D_{-1}	X	X	D_2	D_1	D_0	D_{-1}
L	H	L	X	X	D_1	D_0	D_{-1}	D_{-2}	X	D_1	D_0	D_{-1}	D_{-2}
L	H	H	X	X	X	D_0	D_{-1}	D_{-2}	D_{-3}	D_0	D_{-1}	D_{-2}	D_{-3}

H = HIGH X = DON'T CARE
L = LOW Z = HIGH IMPEDANCE STATE
D_n AT INPUT I_n MAY BE EITHER HIGH OR LOW AND OUTPUT Y_m WILL FOLLOW THE SELECTED D_n INPUT LEVEL.

IC-25S10

26S10 BUS TRANSCEIVER CHIP

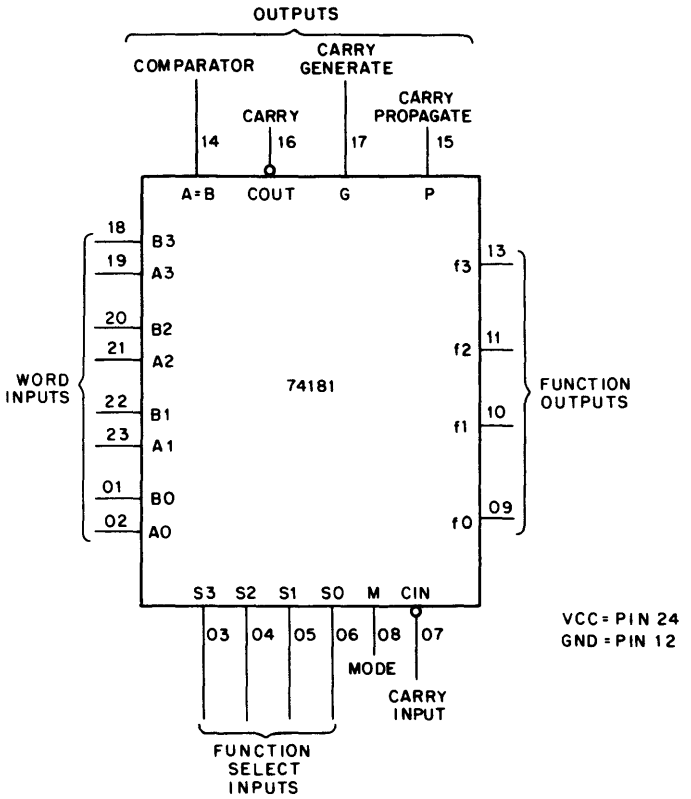


OUTPUTS		INPUTS	
$\overline{E}$	D	$\overline{B}$	Z
L	L	H	L
L	H	L	H
H	X	Y	$\overline{Y}$

H = HIGH
L = LOW
X = DON'T CARE
Y = VOLTAGE OF BUS (ASSUMES CONTROL BY ANOTHER BUS TRANCEIVER)

IC-26S10

74LS181 ALU CHIP



74181
TABLE OF LOGIC FUNCTIONS

Function Select				Output Function	
S3	S2	S1	S0	Negative Logic	Positive Logic
L	L	L	L	$f = \overline{A}$	$f = \overline{A}$
L	L	L	H	$f = \overline{AB}$	$f = \overline{A} + B$
L	L	H	L	$f = \overline{A} + B$	$f = \overline{AB}$
L	L	H	H	f = Logical 1	f = Logical 0
L	H	L	L	$f = \overline{A + B}$	$f = \overline{AB}$
L	H	L	H	$f = \overline{B}$	$f = \overline{B}$
L	H	H	L	$f = A \overline{\odot} B$	$f = A \odot B$
L	H	H	H	$f = A + \overline{B}$	$f = A\overline{B}$
H	L	L	L	$f = \overline{AB}$	$f = \overline{A + B}$
H	L	L	H	$f = A \odot B$	$f = A \overline{\odot} B$
H	L	H	L	f = B	f = B
H	L	H	H	$f = A + B$	f = AB
H	H	L	L	f = Logical 0	f = Logical 1
H	H	L	H	$f = A\overline{B}$	$f = A + \overline{B}$
H	H	H	L	f = AB	$f = A + B$
H	H	H	H	f = A	f = A

With mode control (M) high: C_{in} irrelevant
For positive logic: logical 1 = high voltage
logical 0 = low voltage
For negative logic: logical 1 = low voltage
logical 0 = high voltage

74181
TABLE OF ARITHMETIC OPERATIONS

Function Select				Output Function	
S3	S2	S1	S0	Low Levels Active	High Levels Active
L	L	L	L	f = A minus 1	f = A
L	L	L	H	f = AB minus 1	f = A + B
L	L	H	L	$f = A\overline{B}$ minus 1	$f = A + \overline{B}$
L	L	H	H	f = minus 1 (2's complement)	f = minus 1 (2's complement)
L	H	L	L	f = A plus [A + $\overline{B}$]	$f = A$ plus $A\overline{B}$
L	H	L	H	f = AB plus [A + $\overline{B}$]	f = [A + B] plus $A\overline{B}$
L	H	H	L	f = A minus B minus 1	f = A minus B minus 1
L	H	H	H	$f = A + \overline{B}$	$f = A\overline{B}$ minus 1
H	L	L	L	f = A plus [A + B]	f = A plus AB
H	L	L	H	f = A plus B	f = A plus B
H	L	H	L	f = $A\overline{B}$ plus [A + B]	f = [A + $\overline{B}$] plus AB
H	L	H	H	$f = A + B$	f = AB minus 1
H	H	L	L	f = A plus A†	f = A plus A†
H	H	L	H	f = AB plus A	f = [A + B] plus A
H	H	H	L	f = $A\overline{B}$ plus A	f = [A + $\overline{B}$] plus A
H	H	H	H	f = A	f = A minus 1

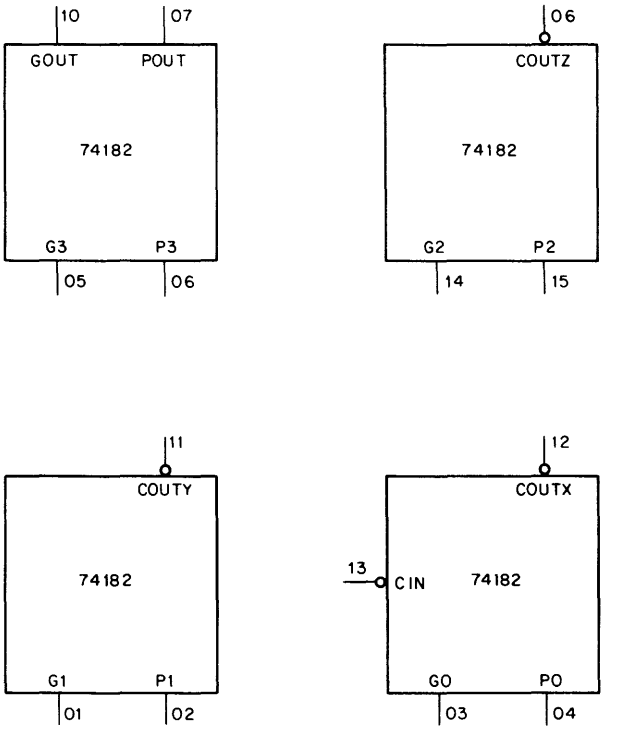
With mode control (M) and C_{in} low
† Each bit is shifted to the next more significant position.

IC-741818

74182 LOOKAHEAD CARRY CHIP

PIN DESIGNATIONS

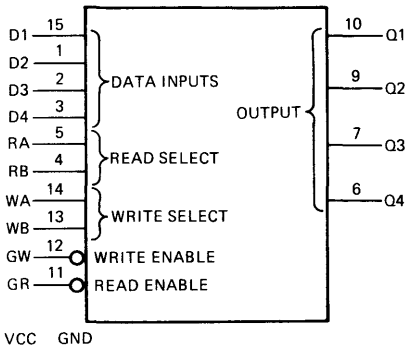
Designation	Pin No.	Function
G0, G1, G2, G3	3, 1, 14, 5	ACTIVE-LOW CARRY GENERATE INPUTS
P0, P1, P2, P3	4, 2, 15, 6	ACTIVE-LOW CARRY PROPAGATE INPUTS
CIN	13	CARRY INPUT
COUTX, COUTY, COUTZ	12, 11, 9	CARRY OUTPUTS
GOUT	10	ACTIVE-LOW CARRY GENERATE OUTPUT
POUT	7	ACTIVE-LOW CARRY PROPAGATE OUTPUT
V_{CC}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



VCC = PIN 16
GND = PIN 08

IC-74182

74LS670 4 X 4 REGISTER FILE CHIP



TRUTH TABLES

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

WRITE INPUTS			WORD			
W_B	W_A	G_W	0	1	2	3
L	L	L	Q=D	Q_0	Q_0	Q_0
L	H	L	Q_0	Q=D	Q_0	Q_0
H	L	L	Q_0	Q_0	Q=D	Q_0
H	H	L	Q_0	Q_0	Q_0	Q=D
X	X	H	Q_0	Q_0	Q_0	Q_0

READ FUNCTION TABLE
(SEE NOTES A AND D)

READ INPUTS			OUTPUTS			
R_B	R_A	G_R	Q1	Q2	Q3	Q4
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)

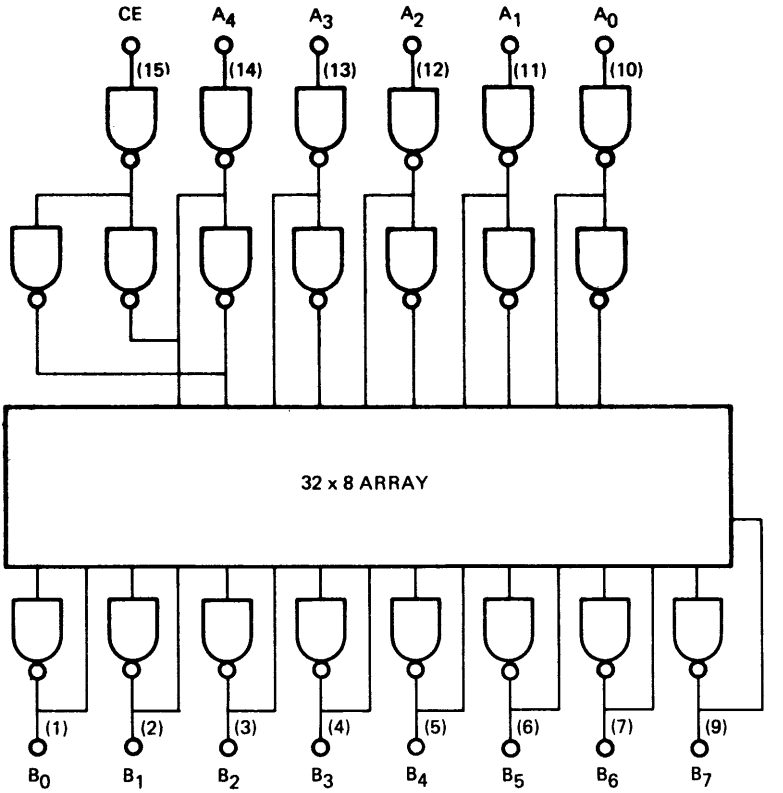
B. (Q = D) = THE FOUR SELECTED INTERNAL FLIP-FLOP OUTPUTS WILL ASSUME THE STATES APPLIED TO THE FOUR EXTERNAL DATA INPUTS.

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

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

IC-74LS670

82S23, 82S123 256-BIT BIPOLAR PROM CHIP

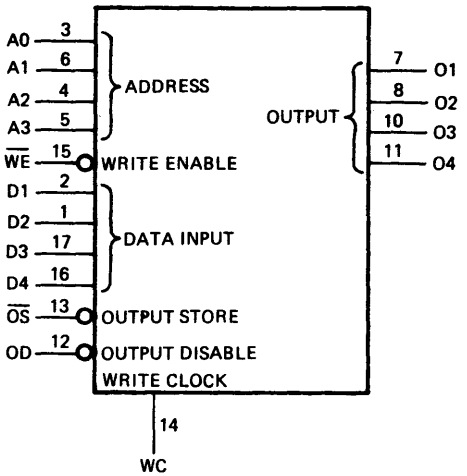


THE 82S23 USER OPEN COLLECTOR OUTPUTS.
THE 82S123 USER TRISTATE OUTPUTS.

V_{CC} = (16)
GND = (8)
(N) = DENOTES PIN NUMBERS

IC-82S23
82S123

85S68 64-BIT EDGE-TRIGGERED D-TYPE REGISTER FILE CHIP WITH TRISTATE OUTPUTS



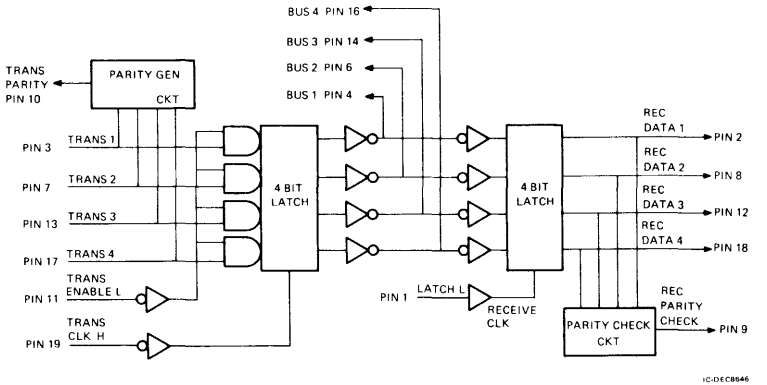
VCC 18
GND 9

TRUTH TABLE

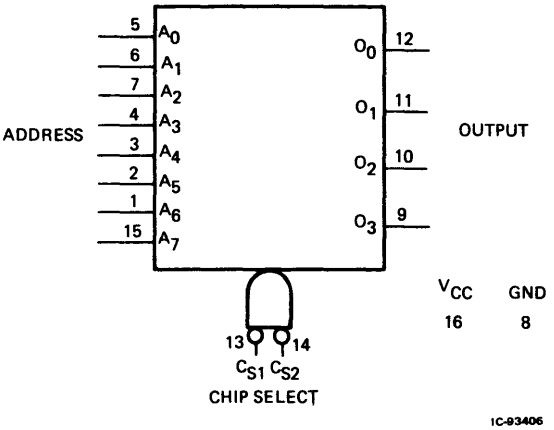
OD	$\overline{WE}$	CLK	$\overline{OS}$	MODE	OUTPUTS
L	X	X	L	OUTPUT STORE	DATA FROM LAST ADDRESSED LOCATION
X	L	⌐	X	WRITE DATA	DEPENDENT ON STATE OF OD AND $\overline{OS}$
L	X	X	H	READ DATA	DATA STORED IN ADDRESSED LOCATION
H	X	X	L	OUTPUT STORE	Hi-Z
H	X	X	H	OUTPUT DISABLE	Hi-Z

IC-85568

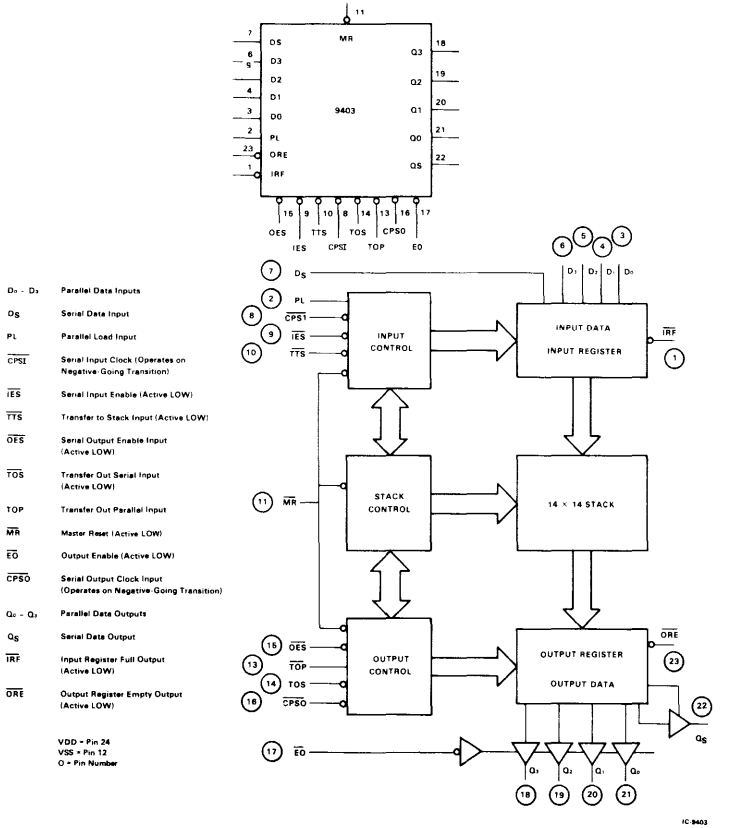
DEC 8646 FOUR-BIT TRISTATE BACKPLANE INTERCONNECT TRANSCEIVER CHIP



93406 1024-BIT ROM CHIP



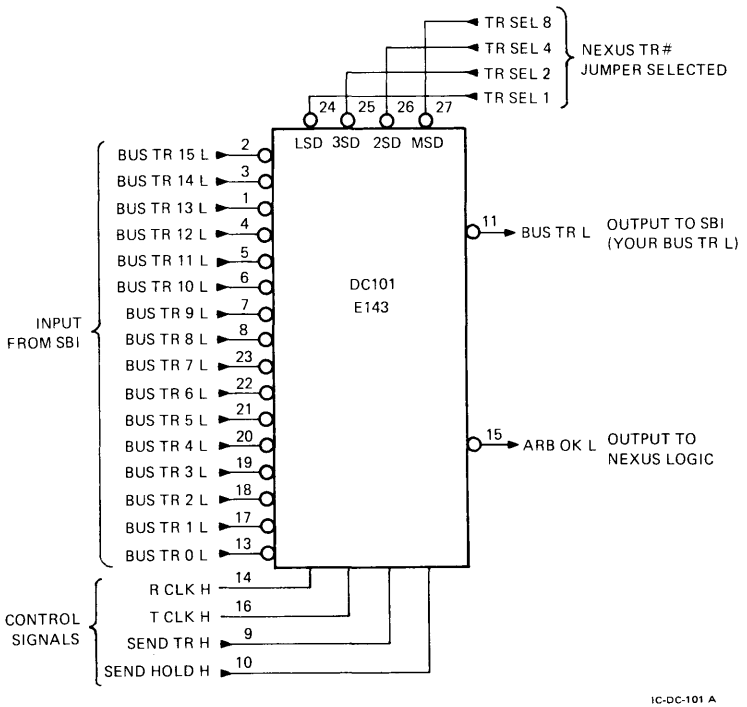
9403 FIFO BUFFER CHIP



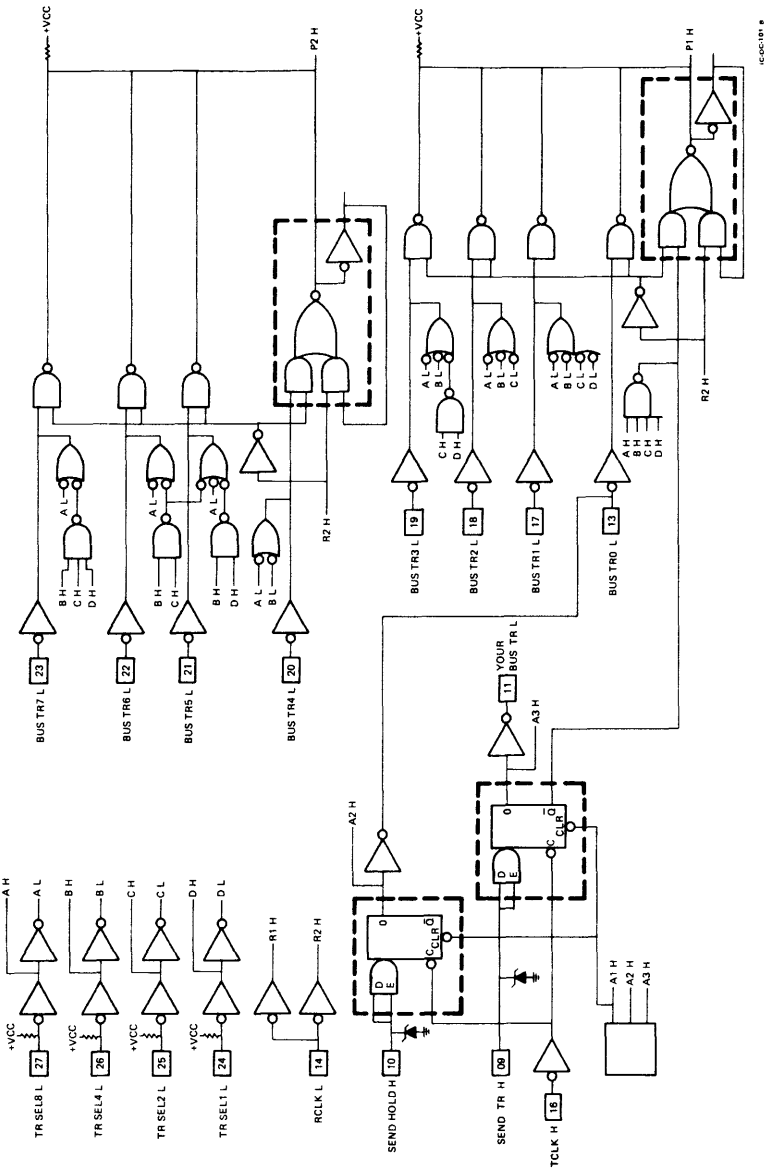
D_0 - D_3	Parallel Data Inputs
D_S	Serial Data Input
PL	Parallel Load Input
$\overline{CPSI}$	Serial Input Clock (Operates on Negative-Going Transition)
$\overline{IES}$	Serial Input Enable (Active LOW)
$\overline{TTS}$	Transfer to Stack Input (Active LOW)
$\overline{OES}$	Serial Output Enable Input (Active LOW)
$\overline{TOS}$	Transfer Out Serial Input (Active LOW)
TOP	Transfer Out Parallel Input
$\overline{MR}$	Master Reset (Active LOW)
$\overline{EO}$	Output Enable (Active LOW)
$\overline{CPSO}$	Serial Output Clock Input (Operates on Negative-Going Transition)
Q_0 - Q_3	Parallel Data Outputs
Q_S	Serial Data Output
$\overline{IRF}$	Input Register Full Output (Active LOW)
$\overline{ORE}$	Output Register Empty Output (Active LOW)

VDD = Pin 24
VSS = Pin 12
O = Pin Number

DC101 ARBITRATOR CHIP, PART 1

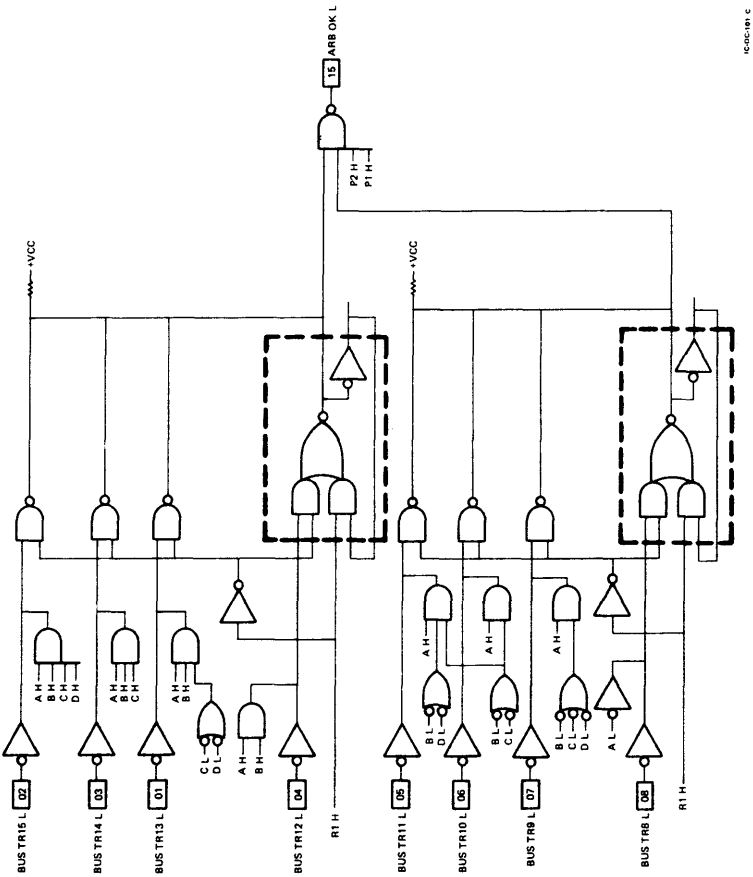


DC101 ARBITRATOR CHIP, PART 2

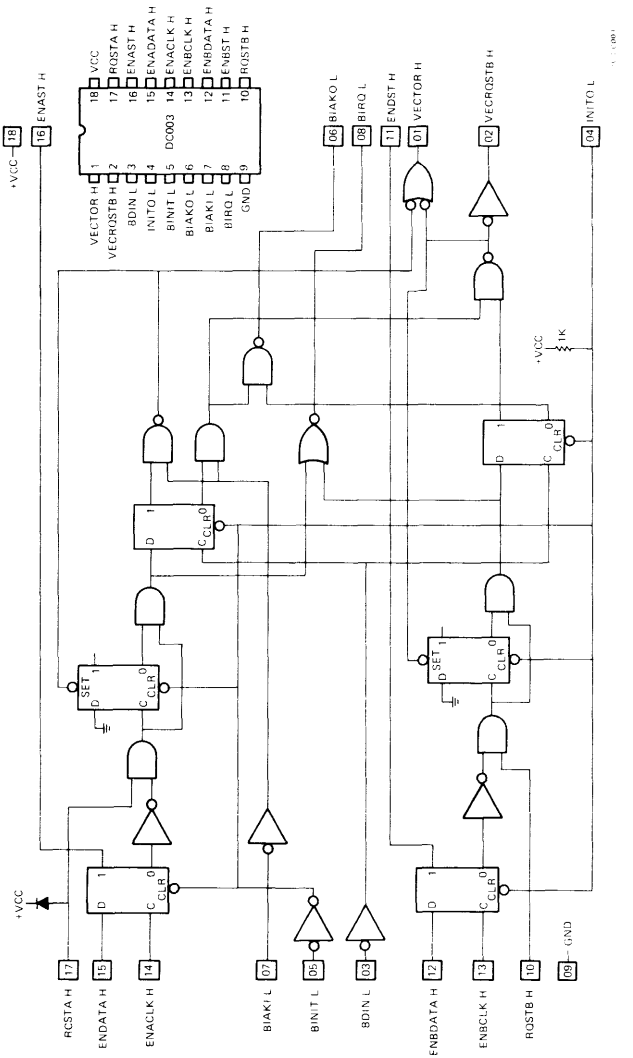


VAX-11/780 INTEGRATED CIRCUIT DIAGRAMS (CONT)

DC101 ARBITRATOR CHIP, PART 3

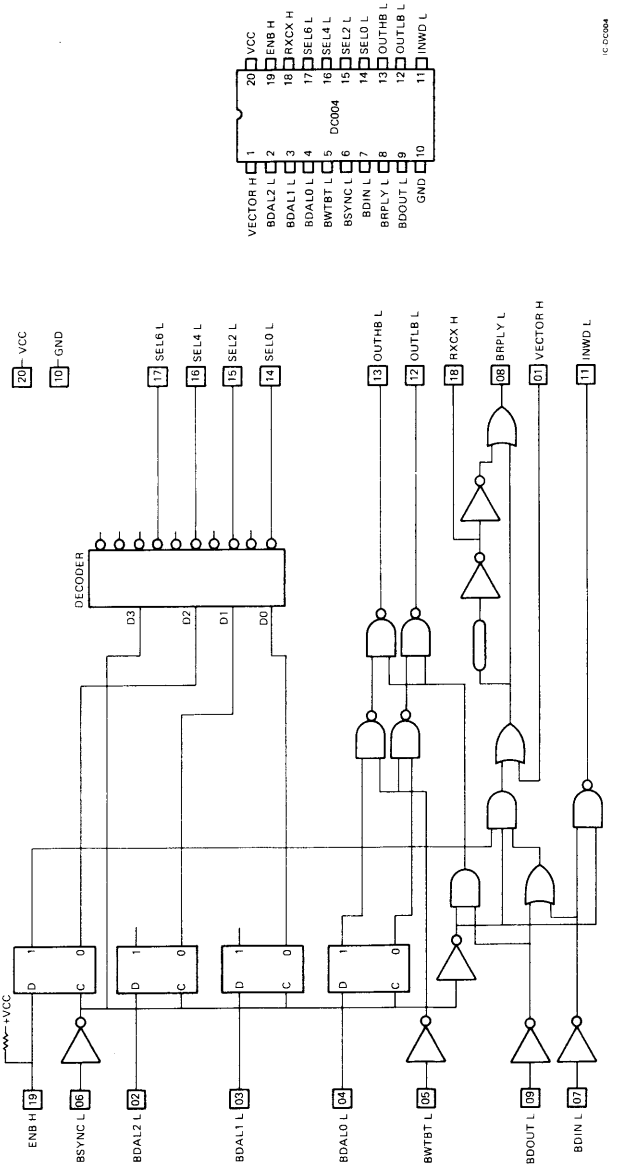


DC003 INTERRUPT CHIP



VAX-11/780 INTEGRATED CIRCUIT DIAGRAMS (CONT)

DC004 PROTOCOL CHIP



DC005 TRANSCEIVER CHIP