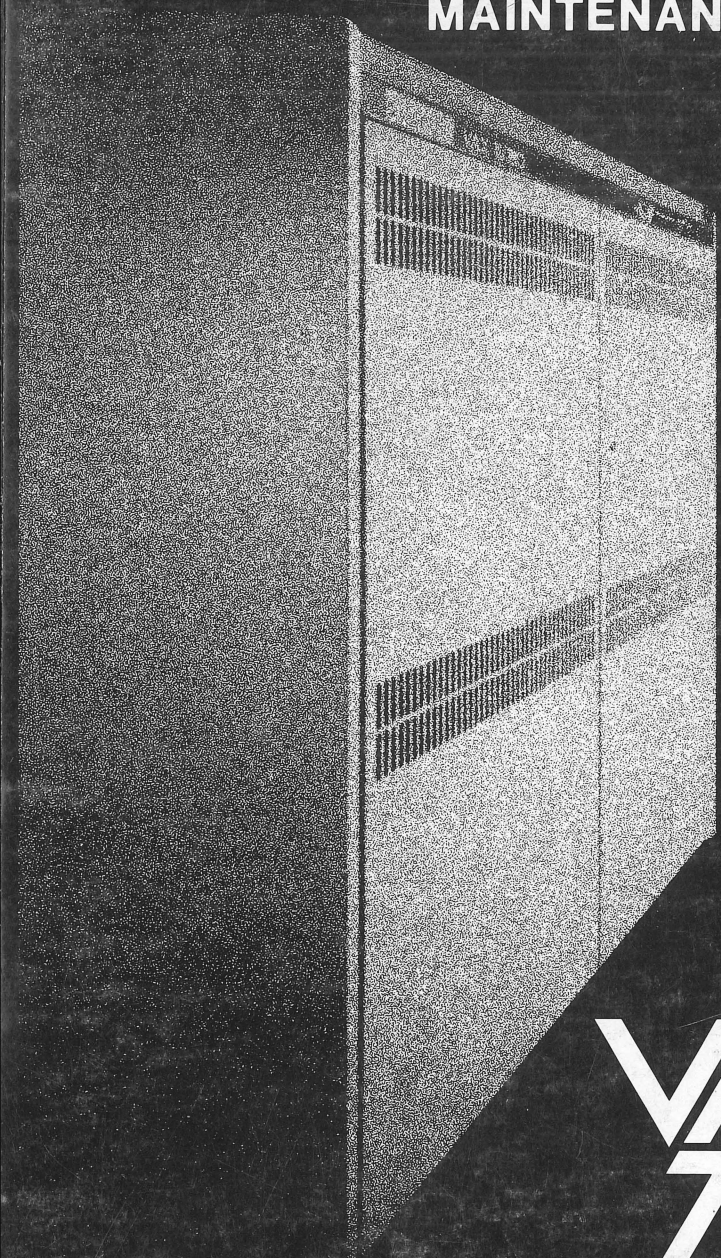
digital

SYSTEM MAINTENANCE GUIDE



VAX11 780

CHRIS MC

EK-11780-PG-001

VAX-11/780
SYSTEM MAINTENANCE GUIDE

First Edition, December 1978

Printed in U.S.A.

CONTENTS

SECTION 1 INTRODUCTION

Introduction 1-3
VAX-11/780 Hardware Manuals 1-4
VAX-11/780 Peripheral Manuals 1-5
VAX-11/780 Software Documentation 1-6

SECTION 2 ARCHITECTURE

Data Types 2-2
Summary of Addressing Modes 2-3
Special Register Usage 2-5
VAX-11 Instruction Set by Opcode 2-6
VAX-11 Instruction Operand Specifier Notation 2-7
VAX-11 Instruction Set 2-8
Branch Conditions 2-26
VAX-11/780 Processor Register Addresses 2-27
VAX-11/780 Processor Register Bit Configurations 2-28
System Control Block 2-36
Process Control Block 2-38
Protection Codes 2-39
Interrupt Priority Requests 2-40
Exception Conditions 2-41
Virtual and Physical Address Relationship 2-42
Virtual and Physical Address Space 2-43
Page Table Formats and Page Table Entry Format 2-44
Example of Page Frame Allocation (Relocation) 2-45
Virtual and Physical Address Formats 2-46
Virtual Pages Mapped to Physical Space 2-47
System Virtual to Physical Address Translation Scheme 2-48
System Virtual to Physical Address Translation, Example 2-49
Process Virtual to Physical Address Translation Scheme 2-50

CONTENTS (Continued)

Process Virtual to Physical Address Translation, Example 2-51
Address Calculation for a TB Hit During a Miss Microtrap 2-52
Address Calculation for a TB Miss During a Miss Microtrap 2-53

SECTION 3 HARDWARE BLOCK DIAGRAMS AND REGISTER BIT CONFIGURATIONS

VAX-11/780 General Block Diagram 3-2
CPU Block Diagram 3-3
Data Path Block Diagram 3-4
Instruction Decode Block Diagram 3-5
Instruction Buffer Block Diagram 3-6
PROM Control Store (PCS) Block Diagram 3-7
Writable Control Store (WCS) Block Diagram 3-8
SBI Control Low Bits (SBL) Block Diagram 3-9
SBI Control High Bits (SBH) Block Diagram 3-10
Cache Data Matrix Block Diagram 3-11
Cache Address Matrix Block Diagram 3-12
Translation Buffer Data Matrix Block Diagram 3-13
Translation Buffer Address Matrix Block Diagram 3-14
Clock Module Block Diagram 3-15
Microsequencer Block Diagram 3-16
Floating-Point Accelerator Block Diagram 3-17
Console Subsystem Configuration 3-18
Console Interface Board Block Diagram 3-19
ID Bus Map 3-20
ID Bus Register Bit Configurations 3-26
QBus Signal Description 3-50
QBus Registers (lower) 3-53
QBus Registers (upper) 3-54
SBI Configuration 3-55
SBI Parity Field Configuration 3-56

CONTENTS (Continued)

SBI Field Description 3-57
SBI I/O Register Addressing 3-59
SBI Information Transfer Formats 3-60
SBI Faults 3-61
SBI Signals, Backplane Pins 3-62
Memory Block Diagram, Part 1 3-63
Memory Block Diagram, Part 2 3-64
Memory I/O Data Logic 3-65
Memory Configuration Register A 3-66
Memory Configuration Register B 3-67
Memory Configuration Register C 3-68
UBA (DW780) Block Diagram 3-69
UBA Address Space and C/A Format 3-70
SBI to Unibus Control Address Translation 3-71
Unibus to SBI Address Translation 3-72
Simplified Flow of Major Control Functions within the UBA 3-73
UBA Registers 3-74
- UBA Configuration Register, Bit Configuration 3-74
- UBA Control Register, Bit Configuration 3-74
- UBA Status Register, Bit Configuration 3-75
- UBA Diagnostic Control Register, Bit Configuration 3-75
- UBA Failed Map Entry Register, Bit Configuration 3-76
- UBA Failed Unibus Address Register, Bit Configuration 3-76
- UBA Buffer Selection Verification Register, Bit Configuration 3-76
- UBA BR Receive Vector Register, Bit Configuration 3-77
- UBA Data Path Register, Bit Configuration 3-77
- UBA Map Register, Bit Configuration 3-77

Unibus Configuration 3-78
Unibus Signal Description 3-79

CONTENTS (Continued)

Standard and Modified Unibus Pin Assignments 3-81
Addresses and Vectors for Unibus Devices 3-83
RK611 Register Contents 3-83
DZ11 Register Contents 3-86
MBA (RH780) Block Diagram 3-87
MBA Register Address Offsets 3-89
MBA Registers 3-90
- MBA Configuration/Status Register, Bit Configuration 3-90
- MBA Control Register, Bit Configuration 3-90
- MBA Status Register, Bit Configuration 3-91
- MBA Virtual Address Register, Bit Configuration 3-91
- MBA Byte Counter Register, Bit Configuration 3-92
- MBA Diagnostic Register, Bit Configuration 3-92
- MBA Map Register, Bit Configuration 3-92

Massbus Disk Drive Register Address Calculation Chart 3-93
RP05/RP06 Register Contents 3-94
RM03 Register Contents 3-95
TM03 Register Contents 3-96
Massbus Signal Cable Pin Assignments 3-98

SECTION 4 CONFIGURATION/JUMPERS

Module Utilization Chart 4-2
KA780 TR, SYS.ID Register Jumpers 4-3
KA780 WCS Jumpers 4-4
MS780 Configuration for REV H Backpanel 4-5
Memory Array Addresses 4-6
Memory Syndrome Bit Decoding Chart 4-7
DW780 (UBA) Backpanel Jumper Configuration 4-8
RH780 (MBA) Backpanel Jumper Configuration 4-9
KD11-F Module Jumper Configuration 4-10
MSV-11B Module Jumper Configuration 4-11

M9400-YE Cable Connections 4-12
DLV11 Jumper Configuration 4-13
DLV11-E Jumper Configuration 4-14

SECTION 5 MICROPROCESSOR

Control Store Field Map 5-2
Microcode Routines which Support Console Software, Starting Addresses 5-3
Microcode Branch Enable Functions 5-4
Microtrap Vectors 5-6
How to Read the Microcode 5-7
VAX-11/780 Microcode, Control ROM Field Definitions 5-10
VAX-11/780 Microcode, Memory Control Functions 5-18
VAX-11/780 System Microcode Macros 5-19
FPA Control ROM Field Definitions 5-44

SECTION 6 TROUBLESHOOTING TOOLS/ DIAGNOSTICS

Console Help File 6-2
Console Abbreviation Rules 6-5
Console-Remote Access Help File 6-7
Microdebugger Help File 6-8
Error Message Help File 6-10
V Bus Channel Configuration 6-13
V Bus Directory 6-14
Explanation of Version Numbers for Console Booting 6-26
VAX-11/780 Microcode Machine Check Error Logout 6-27
Double Error Halt 6-29
Microdiagnostics Run, Console Terminal Output 6-30

CONTENTS (Continued)

Load and Run Stand-Alone Macrodiagnostics (off-line)	6-31
Load and Run Macrodiagnostics Under VMS (on-line)	6-32
Microdiagnostic Monitor Commands	6-33
Microdiagnostic Pseudo Instruction	6-40
Microdiagnostics Control ROM Field Definitions	6-52
Diagnostic Supervisor Commands	6-94

SECTION 7 SYSTEM OPERATION

VMS Boot Procedure	7-2
Use of Filex to Transfer Diagnostic Files	7-5
Terminal Function Keys	7-7
Commands for Terminal Communication and Control	7-8
Commands for File Manipulation	7-9
Commands for Device Handling	7-11
Commands for Program Development and Control	7-12
Commands for Command Procedures and Batch Jobs	7-14
UETP Operating Instruction Summary	7-16
Printing the Error Log File	7-20

SECTION 8 CONVERSION TABLES AND INTEGRATED CIRCUIT DIAGRAMS

Hex Adder	8-2
Hex Subtracter	8-3
Hex/Decimal Conversion	8-4
Hex/Octal Conversion	8-5
Octal/Decimal Conversion	8-6
Hexadecimal/ASCII Conversion	8-7
25S10 Four Bit Shifter Chip with Tristate Output	8-8
26S10 Bus Transceiver Chip	8-9

CONTENTS (Continued)

74LS181 ALU Chip	8-10
74182 Look Ahead Carry Chip	8-11
74LS670 4 X 4 Register File Chip	8-12
82S23, 82S123 256 Bit Bipolar PROM Chip	8-13
85S68 64 Bit Edge Triggered D Type Register File Chip with Tristate Outputs	8-14
DEC 8646 4 Bit Tristate Backplane Interconnect Transceiver Chip	8-15
93406 1024 Bit ROM Chip	8-16
9403 FIFO Buffer Chip	8-17
DC101 Arbitrator Chip	8-18
DC003 Interrupt Chip	8-21
DC004 Protocol Chip	8-22
DC005 Transceiver Chip	8-23

SECTION 1
INTRODUCTION

INTRODUCTION

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

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

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

Hard copy manuals can be ordered from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532
Attn: Printing and Circulation Services (NR2/M15)
Customer Services Section

For information concerning microfiche libraries, contact:

Digital Equipment Corporation
Micropublishing Group
Crosby Drive
Bedford, MA

VAX-11/780 HARDWARE MANUALS

Document Title	Control Number	Form
VAX-11/780 Power System Technical Description	EK-PS780-TD-001	Fiche
VAX-11/780 System Installation Manual	EK-SI980-IN-001	Hard copy
DS780 Diagnostic System User's Guide	EK-DS780-UG-001	Hard copy
DS780 Diagnostic System Technical Description	EK-DS780-TD-001	Fiche
FP780 Floating Point Processor Technical Description	EK-FP780-TD-001	Fiche
REP05/REP06 Subsystem Technical Documentation	EK-REP06-TD-001	Fiche
VAX-11/780 Central Processor Technical Description	EK-KA780-TD-001	Fiche
VAX-11/780 Memory System Technical Description	EK-MS780-TD-001	Fiche
DW780 Unibus Adaptor Technical Description	EK-DW780-TD-001	Fiche
KC780 Console Interface Technical Description	EK-KC780-TD-001	Fiche
VAX-11/780 Software Handbook	EB08126	Hard copy
VAX-11/780 Architecture Handbook	EB07466	Hard copy

VAX-11/780 PERIPHERAL MANUALS

Document Title	Control Number	Form
LA35/LA35 DECwriter II User's Manual	EK-LA3635-OP-002	Hard copy
VT-52 DECscope Maintenance Manual	EK-VT52-MM-001	Hard copy
RM03 Disk Subsystem User's Manual	EK-RM03-UG-001	Hard copy
RP05/RP06 DEC Disk Storage Drive Technical Manual	ER-0012 67-01/51.20-01	Hard copy
TE15/TE10W/TE10N DECTAPE Transport Maintenance Manual	EK-OTE16-TM-001	Hard copy
RX8/RX11 Floppy Disk Maintenance Manual	EK-RX01-MM-002	Hard copy
LSI-11, PDP-11/03 User's Manual	EK-LSI11-TM-003	Hard copy
LP11/LS11/LA11 Line Printer User's Manual	EK-LPI11-OP-001	Hard copy
CR11/CM11 Card Reader System Manual	EK-CR11-TM-004	Hard copy
LA180 DEC Printer Maintenance Manual	EK-LA180-MM-002	Hard copy
PDP-11 Peripherals Handbook	EB05961	Hard copy

VAX-11/780 SOFTWARE DOCUMENTATION

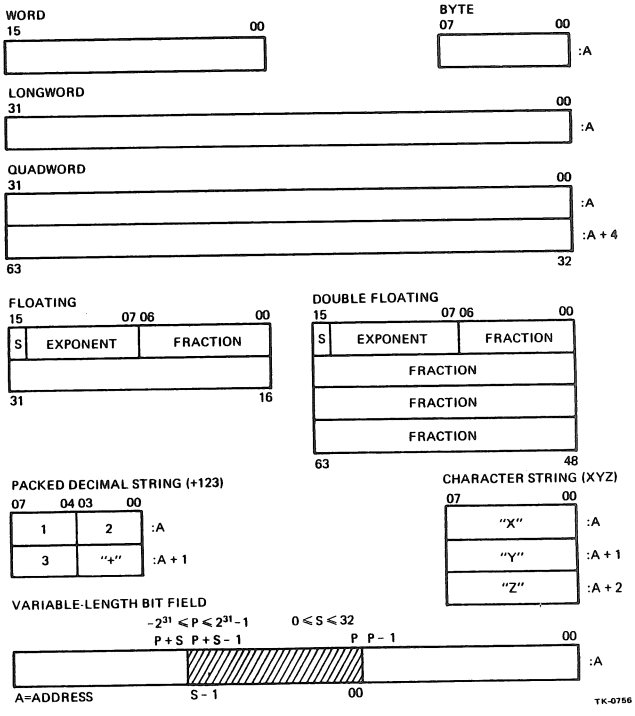
VOLUME	DOCUMENT TITLE	DEC ORDER NUMBER
VOLUME 1A System Reference	VAX/VMS Primer	AA-D030A-TE
	VAX/VMS Summary Description	AA-D022A-TE
	VAX/VMS Information Directory	AA-D016A-TE
	VAX/VMS Release Notes	AA-D015A-TE
	VAX-11 Software Installation Guide	AA-D021A-TE
	VAX/VMS System Services Reference Manual	AA-D018A-TE
VOLUME 1B System Reference	VAX/VMS Command Language User's Guide	AA-D023A-TE
	VAX-11 Linker Reference Manual	AA-D019A-TE
	VAX-11 Symbolic Debugger Reference Manual	AA-D026A-TE
VOLUME 1C System Reference	VAX-11/RSX-11M Programmer's Reference Manual	AA-D020A-TE
	VAX-11/RSX-11M User's Guide	AA-D020A-TE
	VAX-11 MACRO Language Reference Manual	AA-D032A-TE
	VAX-11 MACRO User's Guide	AA-D033A-TE
VOLUME 2A System Procedures	VAX-11 Common Run-time Procedure Library Reference Manual	AA-D036A-TE
	VAX-11 Test Editing Reference Manual	AA-D029A-TE

VOLUME 2B System Procedures	VAX/VMS Operator's Guide	AA-D025A-TE
	VAX/VMS System Manager's Guide	AA-D927A-TE
	VAX/VMS System Messages and Recovery Procedures Manual	AA-D017A-TE
	VAX/VMS UETP User's Guide	AA-D643A-TE
	VAX-11 Disk Save and Compress User's Guide	AA-D739A-TE
VOLUME 3 VAX/VMS I/O	VAX/VMS I/O User's Guide	AA-D028A-TE
	Introduction to VAX-11 Record Management Services	AA-D028A-TE
	VAX-11 Record Management Services Reference Manual	AA-D031A-TE
	VAX-11 Record Management Services User's Guide	AA-D781A-TE
VOLUME 4 RMS-11/ SORT	IAS/RXS-11M RMS-11 MACRO Programmer's Reference Manual	AA-0002A-TC
	Introduction to RMS-11	AA-0001A-TC
	RSX-11M RMS-11 Utilities User's Guide	AA-D083A-TC
	PDP-11 SORT Reference Manual	AA-3341C-TC
VOLUME 5A Optional Software	VAX-11 FORTRAN IV-PLUS Language Reference Manual	AA-D034A-TE
FORTRAN IV-PLUS	VAX-11 FORTRAN IV-PLUS User's Guide	AA-D035A-TE

VOLUME 5B Optional Software	PDP-11 FORTRAN Language Reference Manual	DEC-11-LFLRA-C-D
FORTRAN IV	PDP-11 FORTRAN Language Reference Manual Update Notice No. 1	DEC-11-LFLRA-C-DN1
	IAS-RSX-11 FORTRAN IV User's Guide	DEC-11-LMFUA-D-D
VOLUME 5C Optional Software	PDP-11 COBOL Language Reference Manual	AA-1749D-TC
	PDP-11 COBOL User's Guide	AA-1757C-TC
COBOL	PDP-11 COBOL Pocket Guide	AA-1750C-TC
VOLUME 5D	BASIC PLUS-2 Language Reference Manual	AA-0153A-TK
BASIC PLUS-2	BASIC PLUS-2 RSX-11M/IAS User's Guide	AA-0157A-TC
VOLUME 5E	DECnet-VAX System Manager's	AA-D902A-TE
Optional Software	Guide	
	DECnet-VAX	AA-D901A-TE
DECnet	User's Guide	
VOLUME 5F	User's Guide to	AA-C742A-TC
Optional Software	DATATRIEVE-AA	
DATATRIEVE		

SECTION 2
ARCHITECTURE

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



7 6 5 4 | 3 2 1 0	Dec	Name	Assembler	r m w a v	PC	SP	Indexable?
		GENERAL REGISTER ADDRESSING					
0 0 literal	0-3	literal	S^#literal	y f f f f	-	-	f
4 | Rx	4	indexed	i [Rx]	y y y y y	f	y	f
5 Rn	5	register	Rn	y y y f y	u	uq	f
6 | Rn	6	register deferred	(Rn)	y y y y y	u	y	y
7 Rn	7	autodecrement	-(Rn)	y y y y y	u	y	ux
8 | Rn	8	autoincrement	(Rn)+	y y y y y	p	y	ux
9 Rn	9	autoincrement					
|		deferred	@(R)+	y y y y y	p	y	ux
A Rn	10	byte displacement	B^D (Rn)	y y y y y	p	y	y
B | Rn	11	byte displacement					
		deferred	@B^D (Rn)	y y y y y	p	y	y
C | Rn	12	word displacement	W^D (Rn)	y y y y y	p	y	y
D Rn	13	word displacement					
|		deferred	@W^D (Rn)	y y y y y	p	y	y
E Rn	14	longword displacement	L^D (Rn)	y y y y y	p	y	y
F | Rn	15	longword displacement					
		deferred	@L^D (Rn)	y y y y y	p	y	y

PROGRAM COUNTER ADDRESSING

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

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

SPECIAL REGISTER USAGE

Register	Hardware Use	Conventional Software Use
R0	Results of POLY,CRC; length counter in character & decimal instructions	Results of functions, status of services (not saved or restored on procedure call)
R1	Result of POLYD; address counter in character & decimal instructions	Result of functions (not saved or restored on procedure call)
R2, R4	Length counter in character & decimal instructions	any
R3, R5	Address counter in character & decimal instructions	any
R6-R11	None	any
AP (R12)	Argument pointer saved & loaded by CALL, restored by RET	Argument pointer (base address of argument list)
FP (R13)	frame pointer saved & loaded by CALL, used & restored by RET	Frame pointer; condition signalling
SP (R14)	Stack pointer	Stack pointer
PC (R15)	Program counter	Program counter

LSB / MSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	HALT	NOP	REI	BPT	RET	RSB	LDPCTX	SVPCTX	CVTPS	CVTSP	INDEX	CRC	PROBER	PROBEW	INSQUE	REMQUE
	BSBB	BRB	BNEQ	BEQL	BGTR	BLEQ	JSB	JMP	BGEQ	BLSS	BGTRU	BLEQU	BVC	BVS	BGEQU	BLSSU
1			BNEQU	BEQLU											BCC	BCS
2	ADDP4	ADDP6	SUBP4	SUBP6	CVTPT	MULP	CVTTP	DIVP	MOVC3	CMPC3	SCANC	SPANC	MOVC5	CMPC5	MOVTC	MOVTUC
3	BSBW	BRW	CVTWL	CVTWB	MOVP	CMPP3	CVTPL	CMPP4	EDITPC	MATCHC	LOCC	SKPC	MOVZWL	ACBW	MOVAW	PUSHAW
4	ADDF2	ADDF3	SUBF2	SUBF3	MULF2	MULF3	DIVF2	DIVF3	CVTFB	CVTFW	CVTFL	CVTRFL	CVTBF	CVTWF	CVTLF	ACBF
5	MOVF	CMPF	MNEGF	TSTF	EMODF	POLYF	CVTFD		ADAWI							
6	ADDD2	ADDD3	SUBD2	SUBD3	MULD2	MULD3	DIVD2	DIVD3	CVTDB	CVTDW	CVTDL	CVTRDL	CVTBD	CVTWD	CVTLD	ACBD
	MOVD	CMPD	MNEGD	TSTD	EMODD	POLYD	CVTDF		ASHL	ASHQ	EMUL	EDIV	CLRQ	MOVQ	MOVAQ	PUSHAQ
7													CLRD		MOVAD	PUSHAD
8	ADDB2	ADDB3	SUBB2	SUBB3	MULB2	MULB3	DIVB2	DIVB3	BISB2	BISB3	BICB2	BICB3	XORB2	XORB3	MNEGB	CASEB
9	MOVB	CMPB	MCOMB	BITB	CLRB	TSTB	INCB	DECB	CVTBL	CVTBW	MOVZBL	MOVZBW	ROTL	ACBB	MOVAB	PUSHAB
A	ADDW2	ADDW3	SUBW2	SUBW3	MULW2	MULW3	DIVW2	DIVW3	BISW2	BISW3	BICW2	BICW3	XORWK	XORW3	MNEGW	CASEW
B	MOVW	CMPW	MCOMW	BITW	CLRW	TSTW	INCW	DECW	BISPSW	BICPSW	POPR	PUSHR	CHMR	CHME	CHMS	CHMU
C	ADDL2	ADDL3	SUBL2	SUBL3	MULL2	MULL3	DIVL2	DIVL3	BISL2	BISL3	BICL2	BICL3	XORL2	XORL3	MNEGL	CASEL
	MOVL	CMPL	MCOML	BITL	CLRL	TSTL	INCL	DECL	ADWC	SBWC	MTPR	MFPR	MOVPSL	PUSHL	MOVAL	PUSHAL
D					CLRF										MOVAF	PUSHAF
E	BBS	BBC	BBSS	BBCS	BBSC	BBCC	BBSSI	BBCCI	BLBS	BLBC	FFS	FFC	CMPV	CMPZV	EXTV	EXTZV
F	INSV	ACBL	AOBLSS	AOBLEQ	SOBGEQ	SOBGTR	CVTLB	CVTLW	ASHP	CVTLP	CALLG	CALLS	XFC	ESCD	ESCE	ESCF

VAX-11 INSTRUCTION OPERAND SPECIFIER NOTATION

OPERAND SPECIFIERS ARE SPECIFIED IN THE FOLLOWING MANNER:

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

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

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

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

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

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

   R - OPERAND IS READ ONLY.

   W - OPERAND IS WRITE ONLY.

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

   B - BYTE

   D - DOUBLE FLOATING

   F - FLOATING

   L - LONG WORD

   Q - QUAD WORD

   W - WORD

INSTRUCTIONS	N	CONDITION CODES Z	V	C
9D ACBB ADD COMPARE AND BRANCH BYTE LIMIT.RB, ADD.RB, INDEX.MB, DISPL.BW	INDEX<0	INDEX=0	INTEGER OVERFLOW	C
6F ACBD ADD COMPARE AND BRANCH DOUBLE LIMIT.RD, ADD.RD, INDEX.MD, DISPL.BW	INDEX<0	INDEX=0	FLOATING OVERFLOW	C
4F ACBF ADD COMPARE AND BRANCH FLOATING LIMIT.RF, ADD.RF, INDEX.MF, DISPL.BW	INDEX<0	INDEX=0	FLOATING OVERFLOW	C
F1 ACBL ADD COMPARE AND BRANCH LONG LIMIT.RL, ADD.RL, INDEX.ML, DISPL.BW	INDEX<0	INDEX=0	INTEGER OVERFLOW	C
3D ACBW ADD COMPARE AND BRANCH WORD LIMIT.RW, ADD.RW, INDEX.MW, DISPL.BW	INDEX<0	INDEX=0	INTEGER OVERFLOW	C
58 ADAWI ADD ALIGNED WORD INTERLOCKED ADD.RL, SUM.ML	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
80 ADDB2 ADD BYTE 2 OPERAND ADD.RB, SUM.MB	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
81 ADDB3 ADD BYTE 3 OPERAND ADD1.RB, ADD2.RB, SUM.WB	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
60 ADDD2 ADD DOUBLE 2 OPERAND ADD.RD, SUM.MD	SUM<0	SUM=0	FLOATING OVERFLOW	0
61 ADDD3 ADD DOUBLE 3 OPERAND ADD1.RD, ADD2.RD, SUM.WD	SUM<0	SUM=0	FLOATING OVERFLOW	0
40 ADDF2 ADD FLOATING 2 OPERAND ADD.RF, SUM.MF	SUM<0	SUM=0	FLOATING OVERFLOW	0
41 ADDF3 ADD FLOATING 3 OPERAND ADD1.RF, ADD2.RF, SUM.WF	SUM<0	SUM=0	FLOATING OVERFLOW	0
C0 ADDL2 ADD LONG 2 OPERAND ADD.RL, SUM.ML	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
C1 ADDL3 ADD LONG 3 OPERAND ADD1.RL, ADD2.RL, SUM.WL	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
20 ADDP4 ADD PACKED 4 OPERAND ADDLEN.RW, ADDADDR.AB, SUMLEN.RW, SUMADDR.AB	SUM STRING<0	SUM STRING=0	DECIMAL OVERFLOW	0
21 ADDP6 ADD PACKED 6 OPERAND ADD1LEN.RW, ADD1ADDR.AB, ADD2LEN.RW, ADD2ADDR.AB, SUMLEN.RW, SUMADDR.AB	SUM STRING<0	SUM STRING=0	DECIMAL OVERFLOW	0

INSTRUCTIONS	CONDITION CODES N	Z	V	C
A0 ADDW2 ADD WORD 2 OPERAND ADD.RW, SUM.MW	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
A1 ADDW3 ADD WORD 3 OPERAND ADD.RW, ADD2.RW, SUM.WW	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
D8 ADWC ADD WITH CARRY ADD.RL, SUM.ML	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
F3 AOBLEQ ADD ONE AND BRANCH ON LESS OR EQUAL LIMIT.RL, INDEX.ML, DISPL.BB	INDEX<0	INDEX=0	INTEGER OVERFLOW	C
F2 AOBLSS ADD ONE AND BRANCH ON LESS LIMIT.RL, INDEX.ML, DISPL.BB	INDEX<0	INDEX=0	INTEGER OVERFLOW	C
78 ASHL ARITHMETIC SHIFT LONG CNT.RB, SRC.RL, DST.WL	DST<0	DST=0	INTEGER OVERFLOW	0
F8 ASHP ARTHEMETIC SHIFT AND ROUNDED PACKED CNT.RB, SRCLEN.RW, SRCADDR.AB, ROUND.RB, DSTLEN.RB, DSTADDR.AB	DST STRING<0	DST STRING=0	DECIMAL OVERFLOW	0
79 ASHQ ARITHMETIC SHIFT QUAD CNT.RB, SRC.RQ, DST.WQ	DST<0	DST=0	INT OVERFLOW	0
E1 BBC BRANCH ON BIT CLEAR POS.RL, BASE.VB, DISPL.BB	N	Z	V	C
E5 BBCC BRANCH ON BIT CLEAR AND CLEAR POS.RL, BASE.VB, DISPL.BB	N	Z	V	C
E7 BBCCI BRANCH ON BIT CLEAR AND CLEAR INTERLOCKED POS.RL, BASE.VB, DISPL.BB	N	Z	V	C
E3 BBCS BRANCH ON BIT CLEAR AND SET POS.RL, BASE.VB, DISPL.BB	N	Z	V	C
E0 BBS BRANCH ON BIT SET POS.RL, BASE.VB, DISPL.BB	N	Z	V	C
E4 BBSC BRANCH ON BIT SET AND CLEAR POS.RL, BASE.VB, DISPL.BB	N	Z	V	C
E2 BBSS BRANCH ON BIT SET AND SET POS.RL, BASE.VB, DISPL.BB	N	Z	V	C
E6 BBSS1 BRANCH ON BIT SET AND SET INTERLOCKED POS.RL, BASE.VB, DISPL.BB	N	Z	V	C

INSTRUCTIONS	CONDITION CODES N	Z	V	C
1E BCC BRANCH ON CARRY CLEAR DISPL.BB	N	Z	V	C
1F BCS BRANCH ON CARRY SET DISPL.BB	N	Z	V	C
13 BEQL BRANCH ON EQUAL DISPL.BB	N	Z	V	C
13 BEQLU BRANCH ON EQUAL UNSIGNED DISPL.BB	N	Z	V	C
18 BGEQ BRANCH ON GREATER OR EQUAL DISPL.BB	N	Z	V	C
1E BGEQU BRANCH ON GREATER OR EQUAL UNSIGNED DISPL.BB	N	Z	V	C
14 BGTR BRANCH ON GREATER DISPL.BB	N	Z	V	C
1A BGTRU BRANCH ON GREATER UNSIGNED DISPL.BB	N	Z	V	C
8A BICB2 BIT CLEAR BYTE 2 OPERAND MASK.RB, DST.MB	DST>0	DST=0	0	C
8B BICB3 BIT CLEAR BYTE 3 OPERAND MASK.RB, SRC.RB, DST.WB	DST>0	DST=0	0	C
CA BICL2 BIT CLEAR LONG 2 OPERAND MASK.RL, DST.ML	DST<0	DST=0	0	C
CB BICL3 BIT CLEAR LONG 3 OPERAND MASK.RL, SRC.RL,DST.WL	DST<0	DST=0	0	C
B9 BICPSW BIT CLEAR PROGRAM STATUS WORD MASK.RW	N AND NOT MASK<3>	Z AND NOT MASK<2>	V AND NOT MASK<1>	C AND NOT MASK<0>
AA BICW2 BIT CLEAR WORD 2 OPERAND MASK.RW, DST.MW	DST<0	DST=0	0	C
AB BICW3 BIT CLEAR WORD 3 OPERAND MASK.RW, SRC.RW, DST.WW	DST<0	DST=0	0	C
88 BISB2 BIT SET BYTE 2 OPERAND MASK.RB,DST.MB	DST<0	DST=0	0	C
89 BISB3 BIT SET BYTE 3 OPERAND MASK.RB, SRC.RB, DST.WB	DST<0	DST=0	0	C

INSTRUCTIONS	N	CONDITION CODES Z	V	C
C8 BISL2 BIT SET LONG 2 OPERAND MASK.RL, DST.ML	DST<0	DST=0	0	C
C9 BISL3 BIT SET LONG 3 OPERAND MASK.RL, SRC.RL, DST.WL	DST<0	DST=0	0	C
B8 BISPSW BIT SET PROGRAM STATUS WORD MASK.RW	N OR MASK<3>	Z OR MASK<2>	V OR MASK<1>	C OR MASK<0>
A8 BISW2 BIT SET WORD 2 OPERAND MASK.RW, DST.MW	DST<0	DST=0	0	C
A9 BISW3 BIT SET WORD 3 OPERAND MASK.RW, SRC.RW, DST.WW	DST<0	DST=0	0	C
93 BITB BIT TEST BYTE MASK.RB, SRC.RB	TMP<0	TMP=0	0	C
D3 BITL BIT TEST LONG MASK.RL, SRC.RL	TMP<0	TMP=0	0	C
B3 BITW BIT TEST WORD MASK.RW, SRC.RW	TMP<0	TMP=0	0	C
E9 BLBC BRANCH ON LOW BIT CLEAR SRC.RL, DISPL.BB	N	Z	V	C
E8 BLBS BRANCH ON LOW BIT SET SRC.RL, DISPL.BB	N	Z	V	C
15 BLEQ BRANCH ON LESS OR EQUAL DISPL.BB	N	Z	V	C
1B BLEQU BRANCH ON LESS OR EQUAL UNSIGNED DISPL.BB	N	Z	V	C
19 BLSS BRANCH ON LESS DISPL.BB	N	Z	V	C
1F BLSSU BRANCH ON LESS UNSIGNED DISPL.BB	N	Z	V	C
12 BNEQ BRANCH ON NOT EQUAL DISPL.BB	N	Z	V	C
12 BNEQU BRANCH ON NOT EQUAL UNSIGNED DISPL.BB	N	Z	V	C
03 BPT BREAK POINT TRAP	0	0	0	0

INSTRUCTIONS	N	CONDITION CODES Z	V	C
11 BRB BRANCH WITH BYTE DISPLACEMENT DISPL.BB	N	Z	V	C
31 BRW BRANCH WITH WORD DISPLACEMENT DISPL.BW	N	Z	V	C
10 BSBB BRANCH TO SUBROUTINE WITH BYTE DISPLACEMENT DISPL.BB	N	Z	V	C
30 BSBW BRANCH TO SUBROUTINE WITH WORD DISPLACEMENT DISPL.BW	N	Z	V	C
1C BVC BRANCH ON OVERFLOW CLEAR DISPL.BB	N	Z	V	C
1D BVS BRANCH ON OVERFLOW SET DISPL.BB	N	Z	V	C
FA CALLG CALL WITH GENERAL ARGUMENT LIST ARGLIST.AB, DST.AB	0	0	0	0
FB CALLS CALL WITH STACK NUMARG.RL, DST.AB	0	0	0	0
8F CASEB CASE BYTE SELECTOR.RB, BASE.RB, LIMIT.RL DISPL[0].BW,..., DISPL[LIMIT].BW	TEMP LSS LIMIT	TEMP EQL LIMIT	0	TEMP LSSU LIMIT
CF CASEL CASE LONG SELECTOR.RL, BASE.RL, LIMIT.RL, DISPL[0].BW,..., DISPL[LIMIT].BW	TEMP LSS LIMIT	TEMP EQL LIMIT	0	TEMP LSSU LIMIT
AF CASEW CASE WORD SELECTOR.RW, BASE.RW, LIMIT.RW, DISPL[0].BW,..., DISPL[LIMIT].BW	TEMP LSS LIMIT	TEMP EQL LIMIT	0	TEMP LSSU LIMIT
BD CHME CHANGE MODE TO EXECUTIVE CODE.RW	0	0	0	0
BC CHMK CHANGE MODE TO KERNAL CODE.RW	0	0	0	0
BE CHMS CHANGE MODE TO SUPERVISOR CODE.RW	0	0	0	0
BF CHMU CHANGE MODE TO USER CODE.RW	0	0	0	0

INSTRUCTIONS	CONDITION CODES N	Z	V	C
94 CLRB CLEAR BYTE DST.WB	0	1	0	C
7C CLRD CLEAR DOUBLE DST.WQ	0	1	0	C
D4 CLRF CLEAR FLOAT DST.WL	0	1	0	C
D4 CLRL CLEAR LONG DST.WL	0	1	0	C
7C CLRQ CLEAR QUAD DST.WQ	0	1	0	C
B4 CLRW CLEAR WORD DST.WW	0	1	0	C
91 CMPB COMPARE BYTE SRC1.RB, SRC2.RB	SRC1<SRC2	SRC1=SRC2	0	SRC1 LSSU SRC2 (LSSU=LESS THAN UNSIGNED)
29 CMPC3 COMPARE CHARACTER 3 OPERAND LEN.RW, SRC1ADDR.AB, SRC2ADDR.AB	TER.STR1B <TER.STR2B	TER.STR1B =TER.STR2B	0	TER.STR1B LSSU TER.STR2B
2D CMPC5 COMPARE CHARACTER 5 OPERAND SRC1LEN.RW, SRC1ADDR.AB, FILL.RB, SRC2LEN.RW, SRC2ADDR.AB	TER.STR1B <TER.STR2B	TER.STR1B =TER.STR2B	0	TER.STR1B LSSU TER.STR2B
71 CMPD COMPARE DOUBLE SRC1.RD, SRC2.RD	SRC1<SRC2	SRC1=SRC2	0	0
51 CMPF COMPARE FLOATING SRC1.RF, SRC2.RF	SRC1<SRC2	SRC1=SRC2	0	0
D1 CMPL COMPARE LONG SRC1.RL, SRC2.RL	SRC1<SRC2	SRC1=SRC2	0	SRC1 LSSU SRC2
35 CMPP3 COMPARE PACKED 3 OPERAND LEN.RW, SRC1ADDR.AB, SRC2ADDR.AB (S1S=SOURCE 1 STRING)	S1 STRING<S2 STRING	S1 STRING=S2 STRING	0	0
37 CMPP4 COMPARE PACKED 4 OPERAND SRC1LEN.RW, SRC1ADDR.AB, SRC2LEN.RW, SRC2ADDR.AB (S1S=SOURCE 1 STRING)	S1 STRING<S2 STRING	S1 STRING=S2 STRING	0	0
EC CMPV COMPARE FIELD				

INSTRUCTIONS	CONDITION CODES N	Z	V	C
POS.RL, SIZE.RB, BASE.VB, SRC.RL	TMP<SRC	TMP=SRC	0	TMP LSSU SRC
B1 CMPW COMPARE WORD SRC1.RW, SRC2.RW	SRC1<SRC2	SRC1=SRC2	0	SRC1 LSSU SRC2
ED CMPZV COMPARE ZERO-EXTENDED FIELD POS.RL, SIZE.RB, BASE.VB, SRC.RL	TMP<SRC	TMP=SRC	0	TMP LSSU SRC
0B CRC CALCULATE CYCLIC REDUNDANCY CHECK TBL.AB, INICRC.RL, STRLEN.RW, STREAM.AB, DST.WL	DST<0	DST=0	0	C
6C CVTBD CONVERT BYTE TO DOUBLE SRC.RB, DST.WD	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
4C CVTBF CONVERT BYTE TO FLOAT SRC.RB, DST.WF	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
98 CVTBL CONVERT BYTE TO LONG SRC.RB, DST.WL	DST<0	DST=0	INTEGER OVERFLOW	0
99 CVTBW CONVERT BYTE TO WORD SRC.RB, DST.WW	DST<0	DST=0	INTEGER OVERFLOW	0
68 CVTDB CONVERT DOUBLE TO BYTE SRC.RD, DST.WB	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
76 CVTDF CONVERT DOUBLE TO FLOAT SRC.RD, DST.WF	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
6A CVTDL CONVERT DOUBLE TO LONG SRC.RD, DST.WL	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
69 CVTDW CONVERT DOUBLE TO WORD SRC.RD, DST.WW	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
48 CVTFB CONVERT FLOAT TO BYTE SRC.RF, DST.WB	DST<0	DST=0	SRC CANNOT BE	0

INSTRUCTIONS	N	CONDITION CODES Z	V	C
			REPRESENTED IN DST	
56 CVTFD CONVERT FLOAT TO DOUBLE SRC.RF, DST.WD	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
4A CVTFL CONVERT FLOAT TO LONG SRC.RF, DST.WL	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
49 CVTFW CONVERT FLOAT TO WORD SRC.RF, DST.WW	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
F6 CVTLB CONVERT LONG TO BYTE SRC.RL, DST.WB	DST<0	DST=0	INTEGER OVERFLOW	0
6E CVTLD CONVERT LONG TO DOUBLE SRC.RL, DST.WD	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
4E CVTLF CONVERT LONG TO FLOAT SRC.RL, DST.WF	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
F9 CVTLP CONVERT LONG TO PACKED SRC.RL, DSTLEN.RW, DSTADDR.AB	DST STRING<0	DST STRING=0	DECIMAL OVERFLOW	0
F7 CVTLW CONVERT LONG TO WORD SRC.RL, DST.WW	DST<0	DST=0	INTEGER OVERFLOW	0
36 CVTPL CONVERT PACKED TO LONG SRCLEN.RW, SRCADDR.AB, DST.WL	DST<0	DST=0	INTEGER OVERFLOW	0
08 CVTPS CONVERT PACKED TO LEADING SEPARATE SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB	SRC STRING<0	SRC STRING=0	DECIMAL OVERFLOW	0
24 CVTPT CONVERT PACKED TO TRAILING SRCLEN.RW, SRCADDR.AB, DSTLEN,RW,	DST<0	DST=0	SRC CANNOT BE	0

INSTRUCTIONS	N	CONDITION CODES Z	V	C
DSTADDR.AB			REPRESENTED IN DST	
6B CVTRDL CONVERT ROUNDED DOUBLE TO LONG SRC.RD, DST.WL	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
4B CVTRFL CONVERT ROUNDED FLOAT TO LONG SRC.RF, DST.WL	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
09 CVTSP CONVERT LEADING SEPARATE TO PACKED SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB	DST STRING<0	DST STRING=0	DECIMAL OVERFLOW	0
26 CVTTP CONVERT TRAILING TO PACKED SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB	DST<0	DST=0	DECIMAL OVERFLOW	0
33 CVTWB CONVERT WORD TO BYTE SRC.RW, DST.WB	DST<0	DST=0	INTEGER OVERFLOW	0
6D CVTWD CONVERT WORD TO DOUBLE SRC.RW, DST.WD	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
4D CVTWF CONVERT WORD TO FLOAT SRC.RW, DST.WF	DST<0	DST=0	SRC CANNOT BE REPRESENTED IN DST	0
32 CVTWL CONVERT WORD TO LONG SRC.RW, DST.WL	DST<0	DST=0	INTEGER OVERFLOW	0
97 DECB DECREMENT BYTE DIF.MB	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW INTO MSB
D7 DECL DECREMENT LONG DIF.ML	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW INTO MSB
B7 DECW DECREMENT WORD DIF.MW	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW INTO MSB
86 DIVB2 DIVIDE BYTE 2 OPERAND DIVR.RB, QUO.MB	QUO<0	QUO=0	INTEGER OVERFLOW OR (V← 1 IF DIVR=0)	0

INSTRUCTIONS	CONDITION CODES N	Z	V	C
87 DIVB3 DIVIDE BYTE 3 OPERAND DIVR.RB, DIVD.RB, QUO.WB	QUO<0	QUO=0	INTEGER OVERFLOW OR (V ← 1 IF DIVR=0)	0
66 DIVD2 DIVIDE DOUBLE 2 OPERAND DIVR.RD, QUO.MD	QUO<0	QUO=0	FLOATING OVERFLOW OR (V ← 1 IF DIVR=0)	0
67 DIVD3 DIVIDE DOUBLE 3 OPERAND DIVR.RD, DIVD.RD, QUO.WD	QUO<0	QUO=0	FLOATING OVERFLOW OR (V ← 1 IF DIVR=0)	0
46 DIVF2 DIVIDE FLOATING 2 OPERAND DIVD.RF, QUO.MF	QUO<0	QUO=0	FLOATING OVERFLOW OR (V ← 1 IF DIVR=0)	0
47 DIVF3 DIVIDE FLOATING 3 OPERAND DIVR.RF, DIVD.RF, QUO.WF	QUO<0	QUO=0	FLOATING OVERFLOW OR (V ← 1 IF DIVR=0)	0
C6 DIVL2 DIVIDE LONG 2 OPERAND DIVR.RL, QUO.ML	QUO<0	QUO=0	INTEGER OVERFLOW OR (V ← 1 IF DIVR=0)	0
C7 DIVL3 DIVIDE LONG 3 OPERAND DIVR.RL, DIVD.RL, QUO.WL	QUO<0	QUO=0	INTEGER OVERFLOW OR (V ← 1 IF DIVR=0)	0
27 DIVP DIVIDE PACKED DIVRLEN.RW, DIVRADDR.AB,DIVDLEN.RW DEVDADDR.AB, QUOLEN.RW, QUOADDR.AB	QUO STRING<0	QUO STRING=0	DECIMAL OVERFLOW OR (V ← 1 IF DIVR=0)	0
A6 DIVW2 DIVIDE WORD 2 OPERAND DIVR.RW, QUO.MW	QUO<0	QUO=0	INTEGER OVERFLOW OR (V ← 1 IF DIVR=0)	0
A7 DIVW3 DIVIDE WORD 3 OPERAND DIVR.RW, DIVD.RW, QUO.WW	QUO<0	QUO=0	INTEGER OVERFLOW OR (V ←1 IF DIVR=0)	0
38 EDITPC EDIT PACKED TO CHARACTER STRING SRCLEN.RW, SRCADDR.AB, PATTERN.AB, DSTADDR.AB	SRC STRING<0	SRC STRING=0	DECIMAL OVERFLOW	SIGNIFICANCE
7B EDIV EXTENDED DIVIDE DIVR.RL, DIVD.RQ, QUO.WL, REM.WL	QUO<0	QUO=0	INTEGER OVERFLOW (V ←1 IF DIVR=0)	0
74 EMODD EXTENDED MULTIPLY AND INTEGERIZE				

INSTRUCTIONS	N	CONDITION CODES Z	V	C
MULR.RD, MULRX.RB, MULD.RD, INT.WL, FRACT.WD	FRACT<0	FRACT=0	INTEGER OVERFLOW	0
54 EMODF EXTENDED MULTIPLY AND INTERGERIZE MULR.RF, MULRX.RB, MULD.RF, INT.WL, FRACT.WF	FRACT<0	FRACT=0	INTEGER OVERFLOW	0
7A EMUL EXTENDED MULTIPLY MULR.RL, MULD.RL, ADD.RL, PROD.WQ	PROD<0	PROD=0	0	0
EE EXTV EXTRACT FIELD POS.RL, SIZE.RB, BASE.VB, DST.WL	DST<0	DST=0	0	0
EF EXTZV EXTRACT ZERO-EXTENDED FIELD POS.RL, SIZE.RB, BASE.VB, DST.WL	DST<0	DST=0	0	0
EB FFC FIND FIRST CLEAR BIT STARTPOS.RL, SIZE.RB, BASE,VB, FINDPOS.WL	0	BIT NOT FOUND	0	0
EA FFS FIND FIRST SET BIT STARTPOS.RL, SIZE.RB, BASE.VB, FINDPOS.WL	0	BIT NOT FOUND	0	0
00 HALT HALT (PRIV INST FAULT/PROCESSOR HALT)	0/N	0/Z	0/V	0/C
96 INCB INCREMENT BYTE SUM.MB	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
D6 INCL INCREMENT LONG SUM.ML	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
B6 INCW INCREMENT WORD SUM.MW	SUM<0	SUM=0	INTEGER OVERFLOW	C FROM MSB
0A INDEX INDEX CALCULATION SUBSCRIPT.RL, LOW.RL, HIGH.RL, SIZE.RL, INDEXIN.RL, INDEXOUT.WL	INDEXOUT<0	INDEXOUT=0	0	0
F0 INSV INSERT FIELD SRC.RL, POS.RL, SIZE.RB, BASE.VB	0	0	0	C
0E INSQUE INSERT INTO QUEUE ENTRY.AB, PRED.AB	ENTRY<ENTRY+4	ENTRY=ENTRY+4	0	ENTRY LSSU ENTRY+4
17 JMP JUMP DST.AB	N	Z	V	C
16 JSB JUMP TO SUBROUTINE				

INSTRUCTIONS	N	CONDITION CODES Z	V	C
DST.AB	N	Z	V	C
06 LDPCTX LOAD PROGRAM CONTEXT	N	Z	V	C
3A LOCC LOCATE CHARACTER CHAR.RB, LEN.RW, ADDR.AB	0	R0=0	0	0
39 MATCHC MATCH CHARACTERS OBJLEN1.RW, OBJADDR1.AB, SRCLEN2.RW, SRCADDR2.AB	0	R0=0	0	0
92 MCOMB MOVE COMPLEMENTED BYTE SRC.RB, DST.WB	DST<0	DST=0	0	C
D2 MCOML MOVE COMPLEMENTED LONG SRC.RL, DST.WL	DST<0	DST=0	0	C
B2 MCOMW MOVE COMPLEMENTED WORD SRC.RW, DST.WW	DST<0	DST=0	0	C
DB MFPR MOVE FROM PROCESSOR REGISTER SRC.RL, DST.WL	DST<0	DST=0	0	C (DST REG REPLACED)
	N	Z	V	C (DST REG NOT REPLACED)
8E MNEGB MOVE NEGATED BYTE SRC.RB, DST.WB	DST<0	DST=0	INTEGER OVERFLOW	DST≠0
72 MNEGD MOVE NEGATED DOUBLE SRC.RD, DST.WD	DST<0	DST=0	0	0
52 MNEGF MOVE NEGATED FLOATING SRC.RF, DST.WF	DST<0	DST=0	0	0
CE MNEGL MOVE NEGATED LONG SCR.RL, DST.WL	DST<0	DST=0	INTEGER OVERFLOW	DST≠0
AE MNEGW MOVE NEGATED WORD SRC.RW, DST.WW	DST<0	DST=0	INTEGER OVERFLOW	DST≠0
9E MOVAB MOVE ADDRESS OF BYTE SRC.AB, DST.WL	DST<0	DST=0	0	C
7E MOVAD MOVE ADDRESS OF DOUBLE SRC.AQ, DST.WL	DST<0	DST=0	0	C
DE MOVAF MOVE ADDRESS OF FLOAT SRC.AL, DST.WL	DST<0	DST=0	0	C

INSTRUCTIONS	CONDITION CODES N	Z	V	C
DE MOVAL MOVE ADDRESS OF LONG SRC.AL, DST.WL	DST<0	DST=0	0	C
7E MOVAQ MOVE ADDRESS OF QUAD SRC.AQ, DST.WL	DST<0	DST=0	0	C
3E MOVAW MOVE ADDRESS OF WORD SRC.AW, DST.WL	DST<0	DST=0	0	C
90 MOVB MOVE BYTE SRC.RB, DST.WB	DST<0	DST=0	0	C
28 MOVC3 MOVE CHARACTER 3 OPERAND LEN.RW, SRCADDR.AB, DSTADDR.AB SRCLEN=SOURCE LENGTH (DSTLEN=DESTINATION LENGTH)	0	1	0	0
2C MOVC5 MOVE CHARACTER 5 OPERAND SRCLEN.RW, SRCADDR.AB, FILL.RB, DSTLEN.RW, DSTADDR.AB SRCLEN=SOURCE LENGTH (DSTLEN=DESTINATION LENGTH)	SRCLEN<DSTLEN	SRCLEN=DSTLEN	0	SRCLEN LSSU DSTLEN
70 MOVD MOVE DOUBLE SRC.RD, DST.WD SRCLEN=SOURCE LENGTH (DSTLEN=DESTINATION LENGTH)	DST<0	DST=0	0	C
50 MOVF MOVE FLOAT SRC.RF, DST.WF	DST<0	DST=0	0	C
D0 MOVL MOVE LONG SRC.RL, DST.WL	DST<0	DST=0	0	C
34 MOVP MOVE PACKED LEN.RW, SRCADDR,AB. DSTADDR.AB	DST<0	DST=0	0	C
DC MOVPSL MOVE PROGRAM STATUS LONGWORD DST.WL	N	Z	V	C
7D MOVQ MOVE QUAD SRC.RQ, DST.WQ	DST STRING<0	DST STRING=0	0	C
2E MOVTC MOVE TRANSLATED CHARACTERS SRCLEN.RW, SRCADDR.AB, FILL.RB, TBLADDR.AB, DSTLEN.RW, DSTADDR.AB	SRCLEN<DSTLEN	SRCLEN=DSTLEN	0	SRCLEN LSSU DSTLEN
2F MOVTUC MOVE TRANSLATED UNTIL CHARACTER SRCLEN.RW, SRCADDR.AB, ESC.RB, TBLADDR.AB, DSTLEN.RW, DSTADDR.AB	SRCLEN<DSTLEN	SRCLEN=DSTLEN	TERMINATED BY ESCAPE	SRCLEN LSSU DSTLEN

INSTRUCTIONS	CONDITION CODES N	Z	V	C
B0 MOVW MOVE WORD SRC.RW, DST.WW	DST<0	DST=0	0	C
9A MOVZBL MOVE ZERO-EXTENDED BYTE TO LONG SRC.RB, DST.WL	0	DST=0	0	C
9B MOVZBW MOVE ZERO-EXTENDED BYTE TO WORD SRC.RB, DST.WW	0	DST=0	0	C
3C MOVZWL MOVE ZERO-EXTENDED WORD TO LONG SRC.RW, DST.WL	0	DST=0	0	C
DA MTPR MOVE TO PROCESSOR REGISTER SRC.RL, DST.RL	SRC<0	SRC=0	0	C (IF REG IS REPLACED)
	N	Z	V	C (IF REG NOT REPLACED)
84 MULB2 MULTIPLY BYTE 2 OPERAND MULR.RB, PROD.WD	PROD<0	PROD=0	INTEGER OVERFLOW	0
85 MULB3 MULTIPLY BYTE 3 OPERAND MULR.RB, MULD.RB, PROD.WB	PROD<0	PROD=0	INTEGER OVERFLOW	0
64 MULD2 MULTIPLY DOUBLE 2 OPERAND MULR.RD, PROD.MD	PROD<0	PROD=0	FLOATING OVERFLOW	0
65 MULD3 MULTIPLY DOUBLE 3 OPERAND MULR.RD, MULR.RD, PROD.WD	PROD<0	PROD=0	FLOATING OVERFLOW	0
44 MULF2 MULTIPLY FLOATING 2 OPERAND MULR.RF, PROD.MF	PROD<0	PROD=0	FLOATING OVERFLOW	0
45 MULF3 MULTIPLY FLOATING 3 OPERAND MULR.RF, MULD.RF, PROD.WF	PROD<0	PROD=0	FLOATING OVERFLOW	0
C4 MULL2 MULTIPLY LONG 2 OPERAND MULR.RL, PROD.ML	PROD<0	PROD=0	INTEGER OVERFLOW	0
C5 MULL3 MULTIPLY LONG 3 OPERAND MULR.RL, MULD.RL, PROD.WL	PROD<0	PROD=0	INTEGER OVERFLOW	0
25 MULP MULTIPLY PACKED MULRLEN.RW, MULRADDR.AB, MULDLEN.RW, MULDADDR.AB, PRODLEN.RW, PRODADDR.AB	PROD STRING<0	PROD STRING=0	DECIMAL OVERFLOW	0
A4 MULW2 MULTIPLY WORD 2 OPERAND MULR.RW, PROD.MW	PROD<0	PROD=0	INTEGER OVERFLOW	0
A5 MULW3 MULTIPLY WORD 3 OPERAND				

INSTRUCTIONS	CONDITION CODES N	Z	V	C
MULR.RW, MULD.RW, PROD.WW	PROD<0	PROD=0	INTEGER OVERFLOW	0
01 NOP NO OPERATION	N	Z	V	C
75 POLYD EVALUATE POLYNOMIAL DOUBLE ARG.RD, DEGREE.RW, TBLADDR.AB	R0<0	R0=0	FLOATING OVERFLOW	0
55 POLYF EVALUATE POLYNOMIAL FLOATING ARG.RF, DEGREE.RW, TBLADDR.AB	R0<0	R0=0	FLOATING OVERFLOW	0
BA POPR POP REGISTERS MASK.RW	N	Z	V	C
0C PROBER PROBE READ ACCESS MODE.RB, LEN.RW, BASE.AB	0	IF BOTH ACCESSIBLE THEN 0, ELSE 1	0	0
0D PROBEW PROBE WRITE ACCESS MODE.RB, LEN.RW, BASE.AB	0	IF BOTH ACCESSIBLE THEN 0, ELSE 1	0	0
9F PUSHAB PUSH ADDRESS OF BYTE SRC.AB	(SP)<0	(SP)=0	0	C
7F PUSHAD PUSH ADDRESS OF DOUBLE SRC.AQ	(SP)<0	(SP)=0	0	C
DF PUSHAF PUSH ADDRESS OF FLOAT SRC.AL	(SP)<0	(SP)=0	0	C
DF PUSHAL PUSH ADDRESS OF LONG SRC.AL	(SP)<0	(SP)=0	0	C
7F PUSHAQ PUSH ADDRESS OF QUAD SRC.AQ	(SP)<0	(SP)=0	0	C
3F PUSHAW PUSH ADDRESS OF WORD SRC.AW	(SP)<0	(SP)=0	0	C
DD PUSHL PUSH LONG SRC.RL	SRC<0	SRC=0	0	C
BB PUSHR PUSH REGISTERS MASK.RW	N	Z	V	C
02 REI RETURN FROM EXCEPTION OR INTERRUPT	SAVED PSL <3>	SAVED PSL <2>	SAVED PSL <1>	SAVED PSL <0>
0F REMQUE REMOVE FORM QUEUE				

INSTRUCTIONS	CONDITION CODES N	Z	V	C
ENTRY.AB, ADDR.WL	ENTRY<ENTRY+4	ENTRY=ENTRY+4	ENTRY=ENTRY+4	ENTRY LSSU ENTRY+4
04 RET RETURN FROM CALLED PROCEDURE	TMPL<3>	TMPL<2>	TMPL<1>	TMPL<0>
9C ROTL ROTATE LONG CNT.RB, SCR.RL, DST.WL	DST<0	DST=0	0	C
05 RSB RETURN FROM SUBROUTINE	N	Z	V	C
D9 SBWC SUBTRACT WITH CARRY SUB.RL, DIF.ML	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW FROM MSB
2A SCANC SCAN FOR CHARACTER LEN.RW, ADDR.AB, TBLADDR.AB, MASK.RB	0	R0=0	0	0
3B SKPC SKIP CHARACTER CHAR.RB, LEN.RW, ADDR.AB	0	R0=0	0	0
F4 SOBGEQ SUBTRACT ONE AND BRANCH ON GREATER OR EQUAL INDEX.ML, DISPL.BB	INDEX<0	INDEX=0	INTEGER OVERFLOW	C
F5 SOBGTR SUBTRACT ONE AND BRANCH ON GREATER INDEX.ML, DISPL.BB	INDEX<0	INDEX=0	INTEGER OVERFLOW	C
2B SPANC SPAN CHARACTERS LEN.RW, ADDR.AB, TBLADDR.AB, MASK.RB	0	R0=0	0	0
82 SUBB2 SUBTRACT BYTE 2 OPERAND SUB.RB, DIF.MB	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW FROM MSB
83 SUBB3 SUBTRACT BYTE 3 OPERAND SUB.RB, MIN.RB, DIF.WB	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW FROM MSB
62 SUBD2 SUBTRACT DOUBLE 2 OPERAND SUB.RD, DIF.MD	DIFF<0	DIFF=0	FLOATING OVERFLOW	0
63 SUBD3 SUBTRACT DOUBLE 3 OPERAND SUB.RD, MIN.RD, DIF.WD	DIFF<0	DIFF=0	FLOATING OVERFLOW	0
42 SUBF2 SUBTRACT FLOATING 2 OPERAND SUB.RF, DIF.MF	DIFF<0	DIFF=0	FLOATING OVERFLOW	0
43 SUBF3 SUBTRACT FLOATING 3 OPERAND SUB.RF, MIN.RF, DIF.WF	DIFF<0	DIFF=0	FLOATING OVERFLOW	0
C2 SUBL2 SUBTRACT LONG 2 OPERAND SUB.RL, DIF.ML	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW FROM MSB
C3 SUBL3 SUBTRACT LONG 3 OPERAND				

INSTRUCTIONS	N	CONDITION CODES Z	V	C
SUB.RL, MIN.RL, DIF.WL	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW FROM MSB
22 SUBP4 SUBTRACT PACKED 4 OPERAND SUBLEN.RW, SUBADDR.AB, DIFLEN.RW, DIFADDR.AB	DIFF STRING<0	DIFF STRING=0	DECIMAL OVERFLOW	0
23 SUBP6 SUBTRACT NUMERPACKED 6 OPERAND SUBLEN.RW, SUBADDR.AB, MINLEN.RW, MINADDR.AB, DIFLEN,RW, DIFADDR.AB	DIFF STRING<0	DIFF STRING=0	DECIMAL OVERFLOW	0
A2 SUBW2 SUBTRACT WORD 2 OPERAND SUB.RW, DIF.MW	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW FROM MSB
A3 SUBW3 SUBTRACT WORD 3 OPERAND SUB.RW, MIN.RW, DIF.WW	DIFF<0	DIFF=0	INTEGER OVERFLOW	BORROW FROM MSB
07 SVPCTX SAVE PROCESS CONTEXT	N	Z	V	C
95 TSTB TEST BYTE SRC.RB	SRC<0	SRC=0	0	0
73 TSTD TEST DOUBLE SRC.RD	SRC<0	SRC=0	0	0
53 TSTF TEST FLOAT SRC.RF	SRC<0	SRC=0	0	0
D5 TSTL TEST LONG SRC.RL	SRC<0	SRC=0	0	0
B5 TSTW TEST WORD SRC.RW	SRC<0	SRC=0	0	0
FC XFC EXTENDED FUNCTION CALL	0	0	0	0
8C XORB2 EXCLUSIVE-OR BYTE 2 OPERAND MASK.RB, DST.MB	DST<0	DST=0	0	C
8D XORB3 EXCLUSIVE-OR BYTE 3 OPERAND MASK.RB, SRC.RB, DST.WB	DST<0	DST=0	0	C
CC XORL2 EXCLUSIVE-OR LONG 2 OPERAND MASK.RL, DST.ML	DST<0	DST=0	0	C
CD XORL3 EXCLUSIVE-OR LONG 3 OPERAND MASK.RL, SRC.RL, DST.WL	DST<0	DST=0	0	C
AC XORW2 EXCLUSIVE-OR WORD 2 OPERAND MASK.RW, DST.MW	DST<0	DST=0	0	C

INSTRUCTIONS		N	CONDITION CODES Z	V	C
AD	XORW3 EXCLUSIVE-OR WORD 3 OPERAND MASK.RW, SRC.RW, DST.WW	DST<0	DST=0	0	C
57	* RESERVED TO DEC *				
59	* RESERVED TO DEC *				
5A	* RESERVED TO DEC *				
5B	* RESERVED TO DEC *				
5C	* RESERVED TO DEC *				
5D	* RESERVED TO DEC *				
5E	* RESERVED TO DEC *				
5F	* RESERVED TO DEC *				
77	* RESERVED TO DEC *				
FD ESCD	* RESERVED TO DEC *				
FE ESCE	* RESERVED TO DEC *				
FF ESCF	* RESERVED TO DEC *				

BRANCH CONDITIONS

OPCODE	CONDITIONS
BGTR	N or Z = 0
BLEQ	N or Z = 1
BNEQ	Z = 0
BNEQU	Z = 0
BEQL	Z = 1
BEQLU	Z = 1
BGEQ	N = 0
BLSS	N = 1
BGTRU	C or Z = 0
BLEQU	C or Z = 1
BVC	V = 0
BVS	V = 1
BGEQU	C = 0
BCC	C = 0
BLSSU	C = 1
BCS	

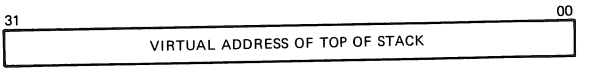
VAX-11/780 PROCESSOR REGISTER ADDRESSES

HEX	DEC			
00	0	KSP	Kernel stack pointer	
01	1	ESP	Executive stack pointer	
02	2	SSP	Supervisor stack pointer	
03	3	USP	User stack pointer	
04	4	ISP	Interrupt stack pointer	
05	5	reserved		
06	6	reserved		
07	7	reserved		
08	8	P0BR	P0 base register	
09	9	P0LR	P0 length register	
0A	10	P1BR	P1 base register	
0B	11	P1LR	P1 length register	
0C	12	SBR	System base register	
0D	13	SLR	System length register	
0E	14	reserved		
0F	15	reserved		
10	16	PCBB	Process control block base	
11	17	SCBB	System control block base	
12	18	IPL	Interrupt priority level	
13	19	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		

VAX-11/780 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**

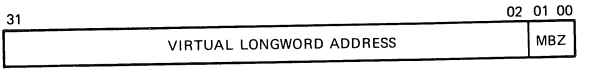


8 08 P0BR 24 **P0 BASE REGISTER**

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**

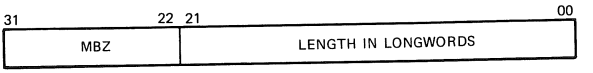
LENGTH OF P0PT IN LONGWORDS

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

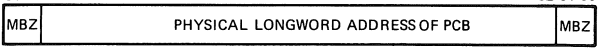


TK-0709

VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

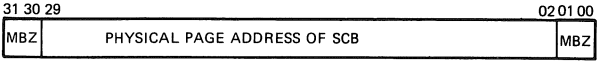
REG. # DEC.	HEX	NAME	ID#	
16	10	PCBB	3A	PROCESS CONTROL BLOCK BASE

RESERVED OPERAND FAULT IF MBZ ≠ 0.

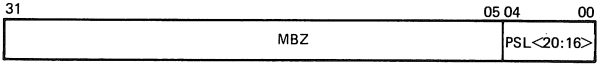


17	11	SCBB	3B	SYSTEM CONTROL BLOCK BASE

RESERVED OPERAND FAULT IF MBZ ≠ 0.

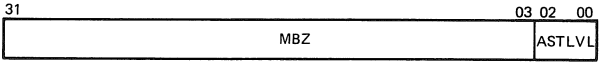


18	12	IPLR	0F	INTERRUPT PRIORITY LEVEL REGISTER



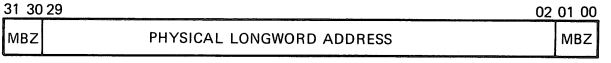
19	13	ASTR	0C	LAST LEVEL REGISTER

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



12	0C	SBR		SYSTEM BASE REGISTER

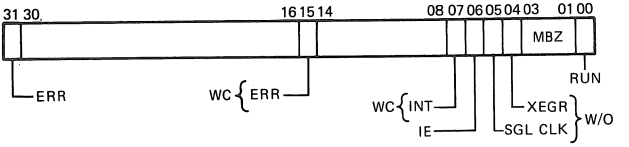
RESERVED OPERAND FAULT IF MBZ ≠ 0.



TK-0711

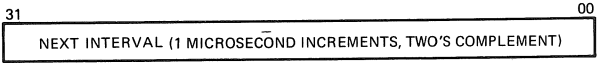
VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

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



BITS 4, 5 ARE 11/780 SPECIFIC

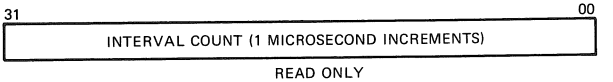
25	19	NICR	09	NEXT INTERVAL COUNT REGISTER



WRITE ONLY

26	1A	ICR	0B	INTERVAL COUNT REGISTER

RESERVED OPERAND FAULT IF WRITE

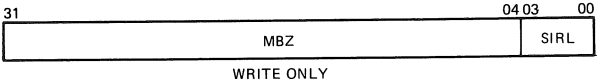


READ ONLY

27	1B	TODR	01	TIME OF DAY REGISTER

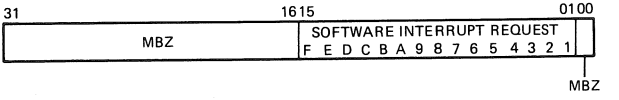
20	14	SIRR		SOFTWARE INTERRUPT REQUEST REGISTER

RESERVED OPERAND FAULT IF READ



WRITE ONLY

21	15	SISR	0E	SOFTWARE INTERRUPT SUMMARY REGISTER

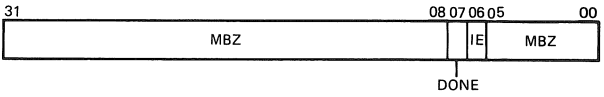


TK-0710

VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

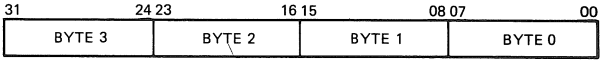
REG.#
DEC HEX NAME ID#

32 20 RXCS 04 CONSOLE RECEIVE CONTROL/STATUS



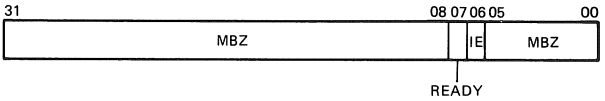
33 21 RXDB 05 CONSOLE RECEIVE DATA BUFFER

RESERVED OPERAND FAULT IF WRITTEN



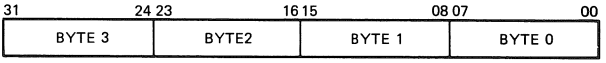
READ ONLY

34 22 TXCS 06 CONSOLE TRANSMIT CONTROL/STATUS



35 23 TXDB 07 CONSOLE TRANSMIT DATA BUFFER

RESERVED OPERAND FAULT IF READ



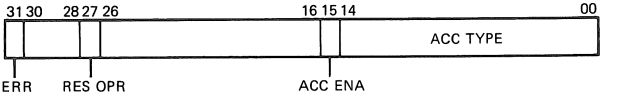
WRITE ONLY

TK-0707

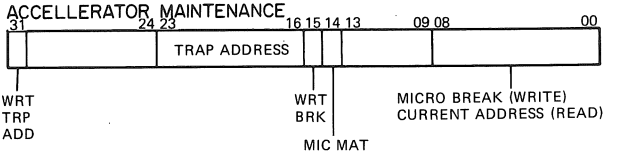
VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

REG.#
DEC HEX NAME ID#

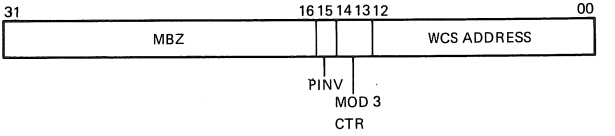
40 28 ACCS 17 ACCELLERATOR CONTROL/STATUS



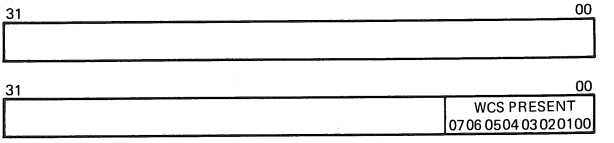
41 29 ACCR 16 ACCELLERATOR MAINTENANCE



44 2C WCSA 22 WRITEABLE CONTROL STORE ADDRESS



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

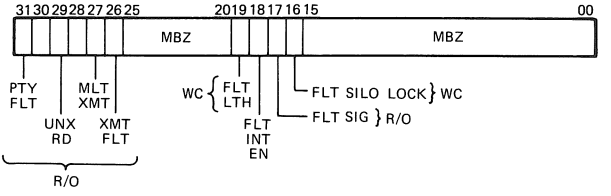


TK-0708

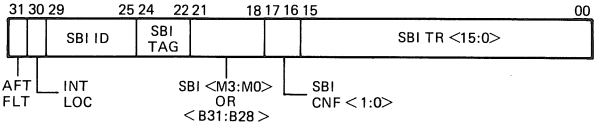
VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

REG.#
DEC HEX NAME ID#

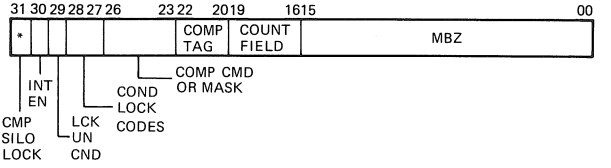
48 30 SBIFS 1B **SBI FAULT/STATUS**



49 31 SBIS 18 **SBI SILO**

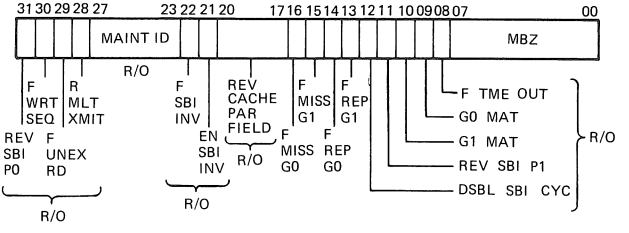


50 32 SBISC 1C **SBI SILO COMPARATOR**



*CLEARED ON ANY WRITE TO SBISC

51 33 SBIMT 1D **SBI MAINTENANCE**

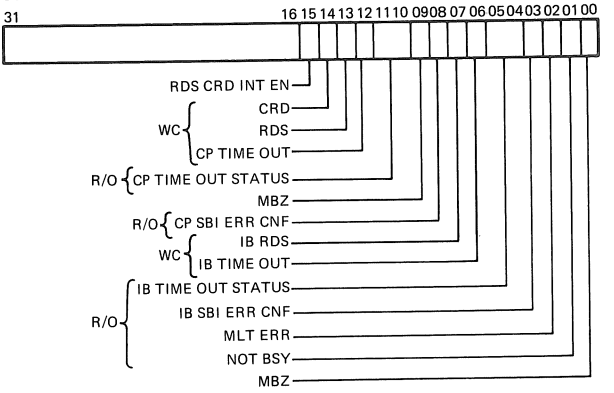


TK-0705

VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

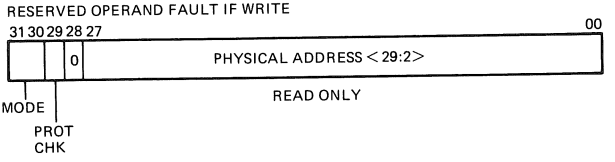
REG.#
DEC HEX NAME ID#

52 34 SBIER 19 **SBI ERROR REGISTER**



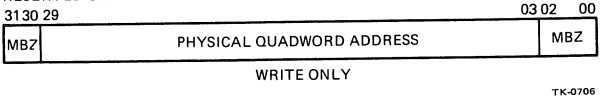
53 35 SBITA 1A **SBI TIMEOUT ADDRESS**

RESERVED OPERAND FAULT IF WRITE



54 36 SBIQC **SBI QUAD CLEAR**

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



TK-0706

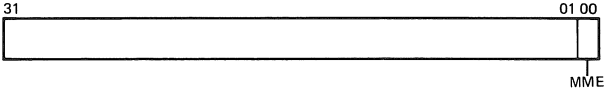
VAX-11/780 PROCESSOR REGISTER BIT CONFIGURATIONS

REG.#
DEC HEX NAME ID#

56 38 MME

MEMORY MANAGEMENT ENABLE

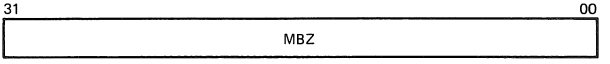
WRITE 1 ALSO CAUSES MICROCODE TO INVALIDATE TB.



57 39 TBIA

TRANSLATION BUFFER INVALIDATE ALL

RESERVED OPERAND FAULT IF READ

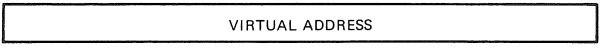


WRITE ONLY

58 3A TBIS

TRANSLATION BUFFER INVALIDATE SINGLE

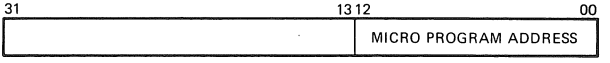
RESERVED OPERAND FAULT IF READ



WRITE ONLY

60 3C MBRK 21

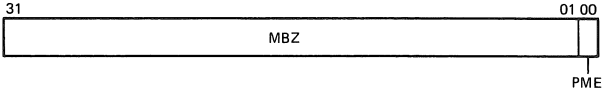
MICROPROGRAM BREAKPOINT



61 3D PMR 0C

PERFORMANCE MONITOR REGISTER

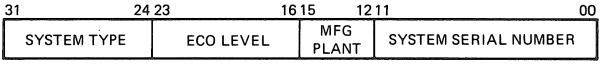
RESERVED OPERAND FAULT IF >1



62 3E SID 03

SYSTEM IDENTIFICATION

RESERVED OPERAND FAULT IF WRITE



READ ONLY

TK-0704

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

SYSTEM CONTROL BLOCK BASE

0	SCBB	MBZ

VECTORS (BITS 1:0)

```
  00   SERVICE ON KERNEL STACK UNLESS RUNNING ON INTERRUPT STACK
  01   SERVICE ON INTERRUPT STACK
**10   SERVICE IN WCS, PASS BITS 15:2 TO MICRO PC
  11   HALT
```

SYSTEM CONTROL BLOCK (SCB)

0	UNUSED, RESERVED		
4	MACHINE CHECK	ABORT/FAULT/TRAP PROCESSOR & ERROR INFO PUSHED ON SP	EXCEPTION VECTOR
8	KERNEL STACK NOT VALID	ABORT VECTOR MUST SELECT IS	EXCEPTION VECTOR
C	POWER FAIL	INTERRUPT	EXCEPTION VECTOR
10	RESERVED/PRIVILEGED INSTRUCTION	FAULT OPCODES RESERVED TO DEC & PRIVILEGED INST.	EXCEPTION VECTOR
14	CUSTOMER RESERVED INSTRUCTION	FAULT	EXCEPTION VECTOR
18	RESERVED OPERAND	FAULT/ABORT	EXCEPTION VECTOR
1C	RESERVED ADDRESSING MODE	FAULT	EXCEPTION VECTOR
20	ACCESS CONTROL VIOLATION	FAULT, VA CAUSING FAULT IS PUSHED ONTO STACK	EXCEPTION VECTOR
24	TRANSLATION NOT VALID	FAULT, VA CAUSING FAULT IS PUSHED ONTO STACK	EXCEPTION VECTOR
28	TRACE (TP)	FAULT ENABLED BY T ON PREVIOUS INSTRUCTION	EXCEPTION VECTOR
2C	BREAKPOINT	FAULT	EXCEPTION VECTOR
30	COMPATIBILITY	TRAP TYPE CODE PUSHED ON STACK (TABLE 1)	EXCEPTION VECTOR
34	ARITHMETIC	TRAP TYPE CODE PUSHED ON STACK (TABLE 2)	EXCEPTION VECTOR
38-3F	UNUSED RESERVED		
40	CHMK	TRAP OPERAND WORD PUSHED ONTO STACK	EXCEPTION VECTOR
44	CHMB	TRAP OPERAND WORD PUSHED ONTO STACK	EXCEPTION VECTOR
48	CHMS	TRAP OPERAND WORD PUSHED ONTO STACK	EXCEPTION VECTOR
4C	CHMU	TRAP OPERAND WORD PUSHED ONTO STACK	EXCEPTION VECTOR
50	SBI SILO COMPARE		INTERRUPT VECTOR
54	CRD/RDS		INTERRUPT VECTOR
*58	SBI ALERT		INTERRUPT VECTOR
*5C	SBI FAULT		INTERRUPT VECTOR
*60	CPU TIMEOUT (VMS: ASYNCHRONOUS WRITE TIMEOUT)		INTERRUPT VECTOR
61-83	UNUSED, RESERVED		
84	SOFTWARE LEVEL 1		INTERRUPT VECTOR
88	SOFTWARE LEVEL 2		INTERRUPT VECTOR
8C-BC	SOFTWARE LEVEL 3-F		INTERRUPT VECTOR
C0	INTERVAL TIMER		INTERRUPT VECTOR
C4-E4	UNUSED, RESERVED		INTERRUPT VECTOR
*F8	CNSL RECEIVE INTR		INTERRUPT VECTOR
*FC	CNSL TRANSMIT INTR		INTERRUPT VECTOR
FF	UNUSED, RESERVED		INTERRUPT VECTOR

*100-13C SBI REQ 4	INTERRUPT VECTOR
*140-17C SBI REQ 5	INTERRUPT VECTOR
*180-1BC SBI REQ 6	INTERRUPT VECTOR
*1C0-1FC SBI REQ 7	INTERRUPT VECTOR

*These offsets are 11/780 specific
**Interrupt serviced in WCS.
Go to 10E0 which contains a RETURN1 unless changed.

VAX-11 Native Mode Codes

CODE	CONDITION
1	INTEGER OVERFLOW TRAP
2	INTEGER DIVIDE BY ZERO TRAP
3	FLOATING OVERFLOW TRAP
4	FLOATING DIVIDE BY ZERO TRAP
5	FLOATING UNDERFLOW TRAP
6	DECIMAL STRING OVERFLOW TRAP
7	DECIMAL STRING DIVIDE BY ZERO TRAP

Compatibility Mode Codes

CODE	CONDITION
0	RESERVED OPCODE
1	BREAKPOINT
2	IOT
3	EMT
4	TRAP
5	ILLEGAL INSTRUCTION
6	ODD ADDRESS

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

PROCESS CONTROL BLOCK BASE (PCB)

PHYSICAL LONG WORD ADDRESS OF PCB	MBZ

PROCESS CONTROL BLOCK (PCB)

				KSP	PCB +0
				ESP	+4
				SSP	+8
				USP	+C
				R0	+10
				R1	+14
				R2	+18
				R3	+1C
				R4	+20
				R5	+24
				R6	+28
				R7	+2C
				R8	+30
				R9	+34
				R10	+38
				R11	+3C
				R12	+40
				R13	+44
				PC	+48
				PSL	+4C
				P0BR	+50
MBZ		ASTLVL	MBZ	P0LR	+54
				P1BR	+58
PME	MBZ			P1LR	+5C

PROTECTION CODES

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

K KERNEL
E EXECUTIVE
S SUPERVISOR
U USER

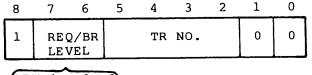
* NO ACCESS
RO READ ONLY
R/W READ WRITE

MODE

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

LEVEL	CONDITION	VECTOR	
IPR 1F	MACHINE CHECK, KERNEL STACK NOT VALID	04, 08	
IPR 1E	CPU POWER FAIL	0C	
IPR 1D	WRITE TIMEOUT	60	
IPR 1C	SBI FAULT	5C	PROCESSOR, MEMORY
IPR 1B	SBI ALERT	58	OR BUS ERROR
IPR 1A	CRD/RDS	54	
IPR 19	SBI SILO COMPARE	50	
IPR 18	INTERVAL TIMER	C0	
IPR 17	SBI REQ7/UNIBUS BR7	1C0-1FC	
IPR 16	SBI REQ6/UNIBUS BR6	180-1BC	
IPR 15	SBI REQ5/UNIBUS BR5	140-17C	DEVICE INTERRUPT
IPR 14	SBI REQ4/UNIBUS BR4	100-13C	
IPR 14	CONSOLE TERM. REC.	F8	
IPR 14	CONSOLE TERM. XMIT.	FC	
IPR 0F	SOFTWARE REQ 0F	BC	
IPR 0E	SOFTWARE REQ 0E	B8	RESERVED
IPR 0D	SOFTWARE REQ 0D	B4	
IPR 0C	SOFTWARE REQ 0C	B0	
IPR 0B	SOFTWARE REQ 0B	AC	
IPR 0A	SOFTWARE REQ 0A	A8	DEVICE DRIVERS
IPR 09	SOFTWARE REQ 09	A4	
IPR 08	SOFTWARE REQ 08	A0	
IPR 07	SOFTWARE REQ 07	9C	TIMER PROCESS
IPR 06	SOFTWARE REQ 06	98	QUEUE ASYNCHRONOUS SYSTEM TRAP (AST)
IPR 05	SOFTWARE REQ 05	94	RESERVED
IPR 04	SOFTWARE REQ 04	90	I/O POST
IPR 03	SOFTWARE REQ 03	8C	PROCESSOR SCHEDULER
IPR 02	SOFTWARE REQ 02 OR AST DEL.	88	AST DELIVERY
IPR 01	SOFTWARE REQ 01	84	RESERVED
IPR 00	SOFTWARE REQ 00		USER PROCESS LEVEL

DEVICE VECTOR GENERATION

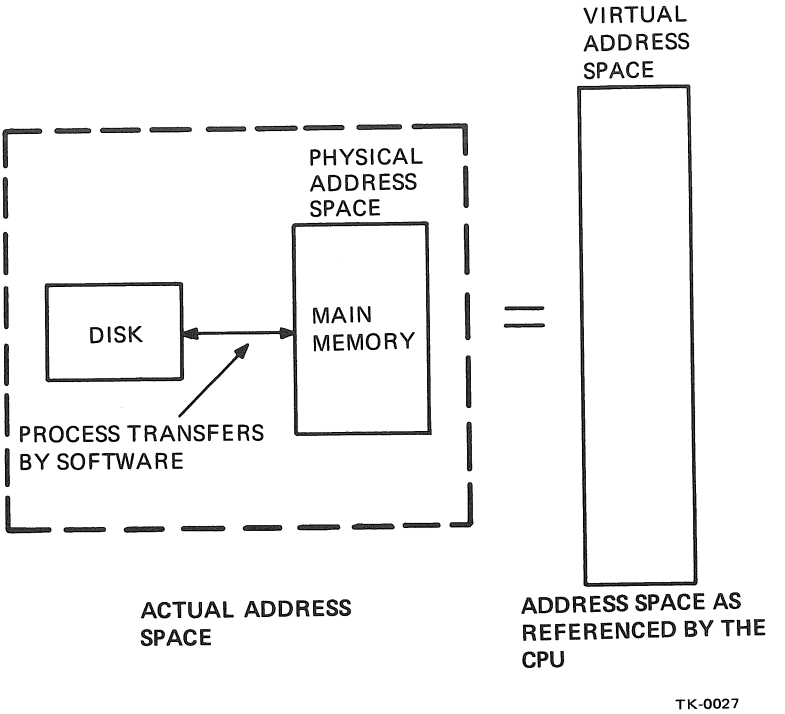


LEV 4 = 0 0
LEV 5 = 0 1
LEV 6 = 1 0
LEV 7 = 1 1

EXCEPTION CONDITIONS

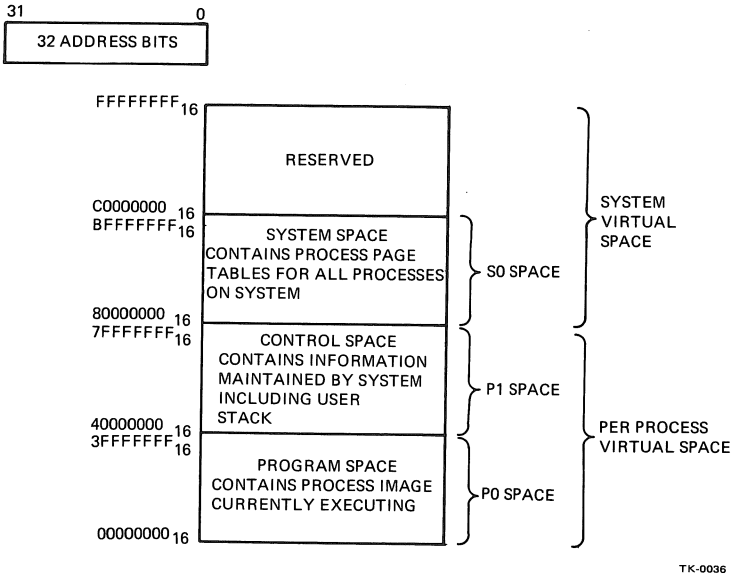
CONDITION	VECTOR
MACHINE CHECK	04
KERNEL STACK NOT VALID	08
RESERVED DEC OPCODES & PRIVILEGED INSTRUCTIONS	10
RESERVED CUSTOMER OPCODES	14
RESERVED OPERANDS	18
RESERVED ADDRESSING MODES	1C
ACCESS CONTROL VIOLATION	20
TRANSLATION NOT VALID	24
TRACE TRAP	28
BPT OPCODE	2C
COMPATABILITY MODE TRAP	30
ARITHMETIC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

VIRTUAL AND PHYSICAL ADDRESS RELATIONSHIP

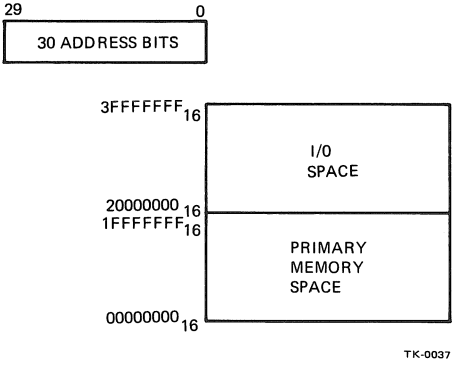


VIRTUAL AND PHYSICAL ADDRESS SPACE

VIRTUAL ADDRESS SPACE



PHYSICAL ADDRESS SPACE



PAGE TABLE FORMATS AND PAGE TABLE ENTRY FORMAT

PAGE TABLE FORMATS

SYSTEM BASE REGISTER
(CONTAINS THE PHYSICAL ADDRESS OF THE FIRST ENTRY OF THE PAGE TABLE)

SYSTEM LENGTH REGISTER
(CONTAINS THE NUMBER OF PAGE TABLE ENTRIES, N)

SYSTEM REGION PAGE TABLE

PAGE TABLE ENTRY FOR VIRTUAL PAGE 0 (FIRST ENTRY)
PTE FOR VPN 1
PTE FOR VPN 2
•
•
•
PAGE TABLE ENTRY FOR VIRTUAL PAGE N - 1 (LAST ENTRY)

PER-PROCESS PAGE TABLES

CONTROL REGION BASE REGISTER
(CONTAINS THE VIRTAUL ADDRESS OF BASE OF THE PAGE TABLE)

CONTROL REGION LENGTH REGISTER
(CONTAINS THE VIRTUAL ADDRESS OF THE FIRST ENTRY IN THE PAGE TABLE FOR VIRTUAL PAGE NUMBER 2**22-N, WHERE N IS THE NUMBER OF PAGE TABLE ENTIRES)

SYSTEM REGION PAGE TABLE

PAGE TABLE ENTRY FOR VIRTUAL PAGE 2*22-N
PTE FOR VPN 2*22-(N-1)
PTE FOR VPN 2*22-(N-2)
PTE FOR VPN 2*22-(N-3)
•
•
•
PTE FOR VPN 2*22-1 (LAST ENTRY)

PROGRAM REGION BASE REGISTER
(CONTAINS THE VIRTUAL ADDRESS OF THE FIRST ENTRY IN THE PAGE TABLE)

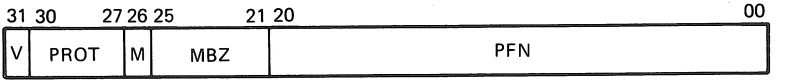
PROGRAM REGION LENGTH REGISTER
(CONTAINS THE NUMBER OF PAGE TABLE ENTRIES, N)

SYSTEM REGION PAGE TABLE

PAGE TABLE ENTRY FOR VIRTUAL PAGE 0 (FIRST ENTRY)
PTE FOR VPN 1
PTE FOR VPN 2
PTE FOR VPN 3
•
•
•
PTE FOR VIRTUAL PAGE N-1 (LAST ENTRY)

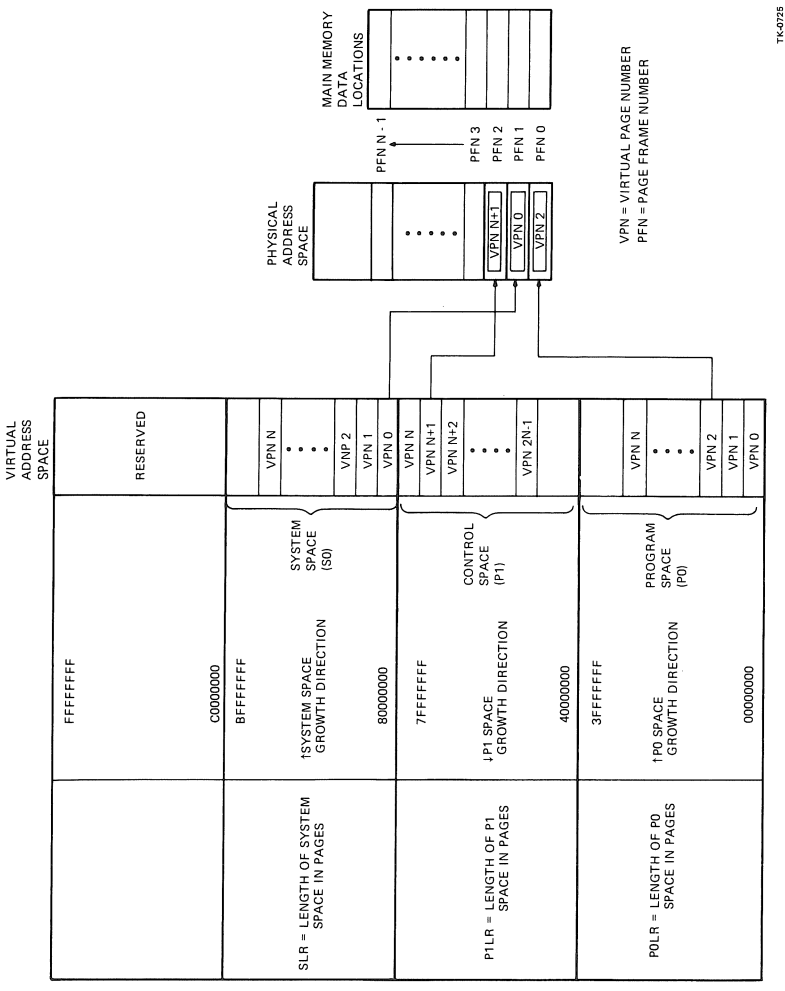
TK-0732

PAGE TABLE ENTRY FORMAT

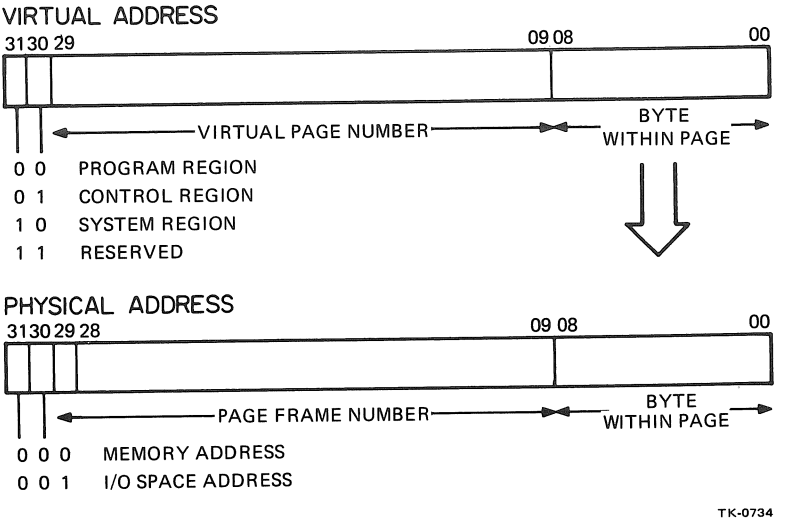


TK-0714

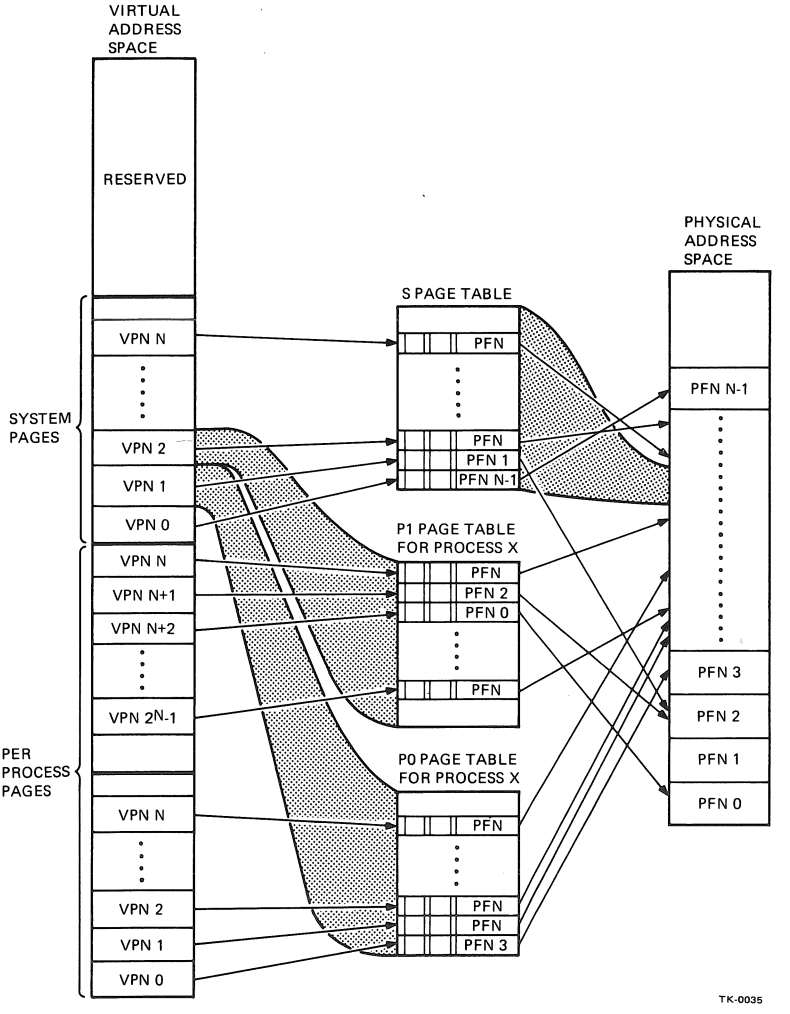
EXAMPLE OF PAGE FRAME ALLOCATION (RELOCATION)



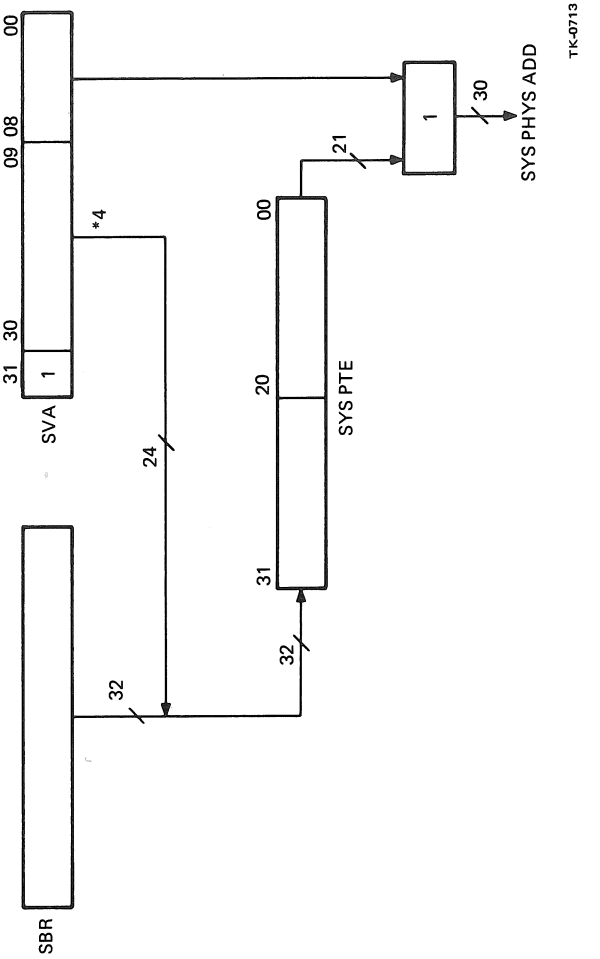
VIRTUAL AND PHYSICAL ADDRESS FORMATS



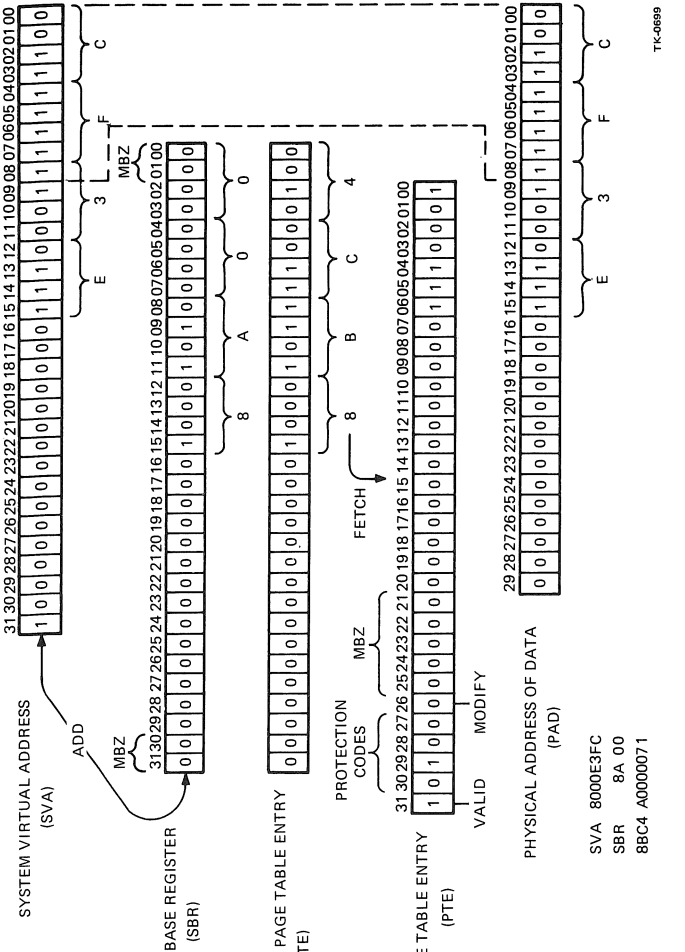
VIRTUAL PAGES MAPPED TO PHYSICAL SPACE



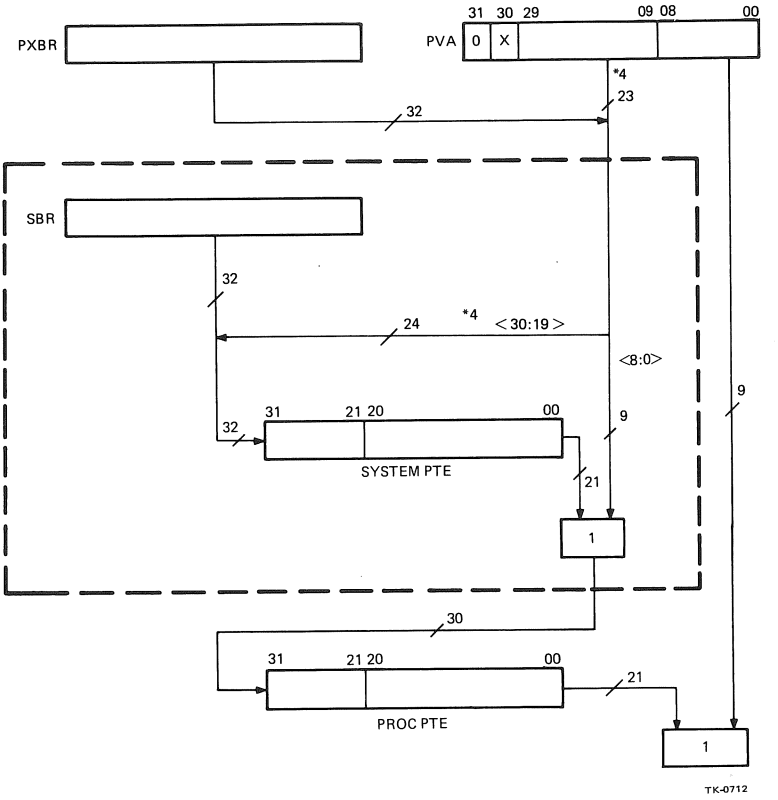
SYSTEM VIRTUAL TO PHYSICAL ADDRESS TRANSLATION SCHEME



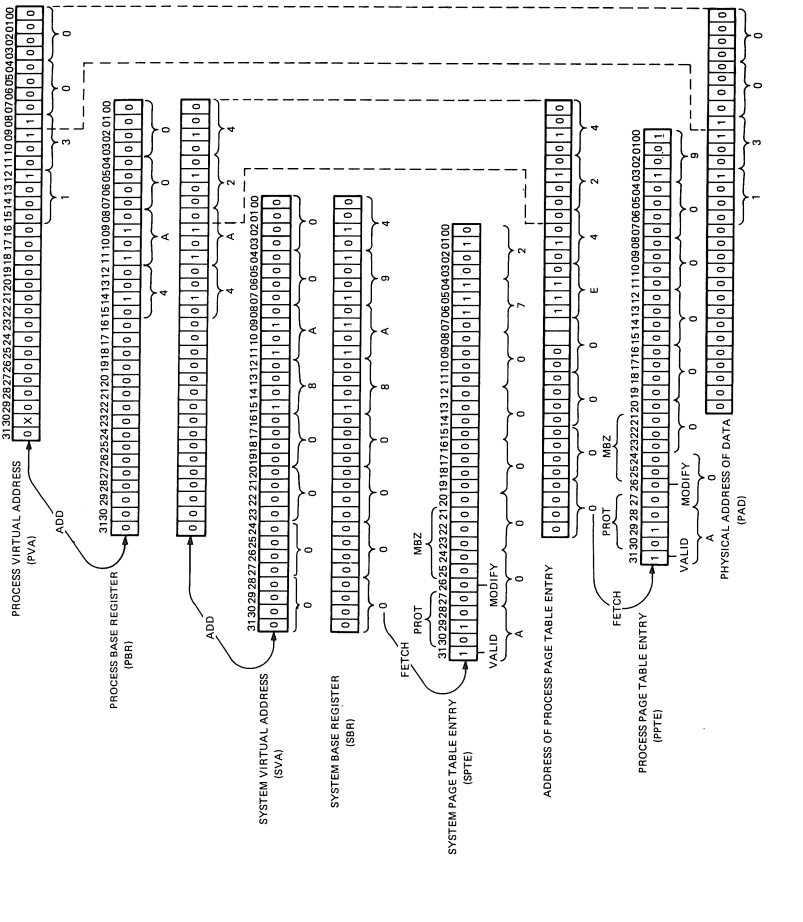
SYSTEM VIRTUAL TO PHYSICAL ADDRESS TRANSLATION, EXAMPLE



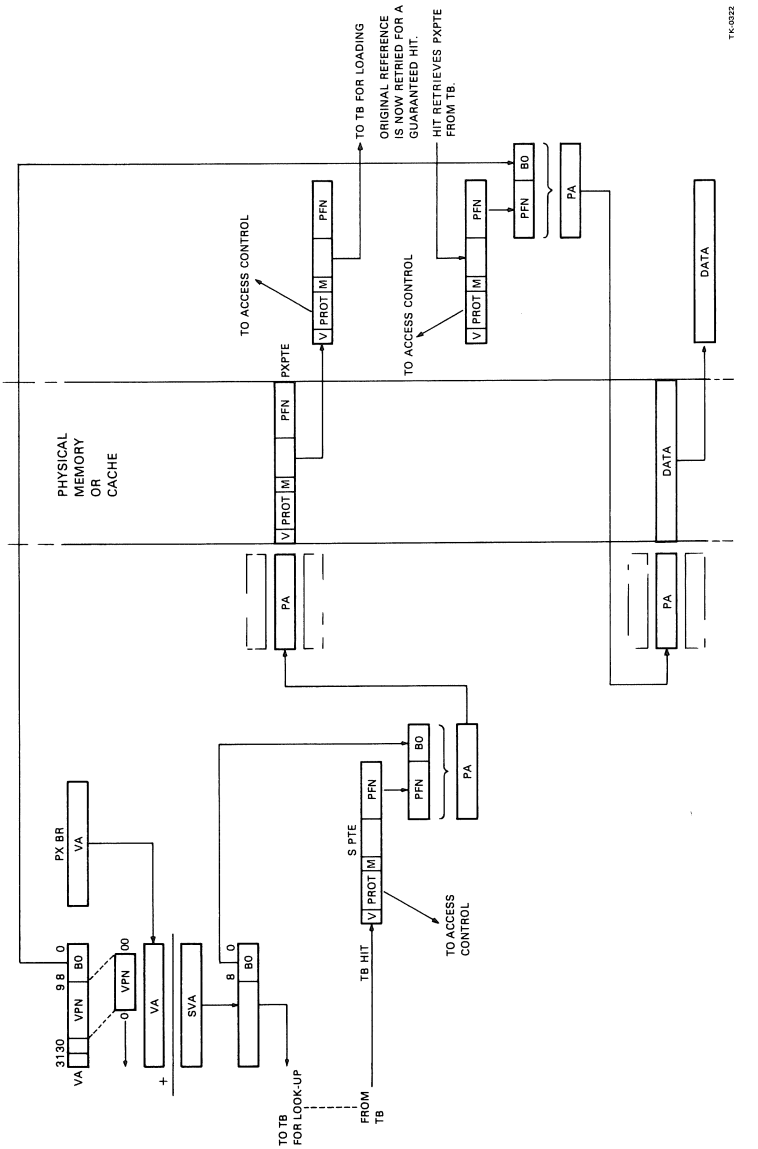
PROCESS VIRTUAL TO PHYSICAL ADDRESS TRANSLATION SCHEME



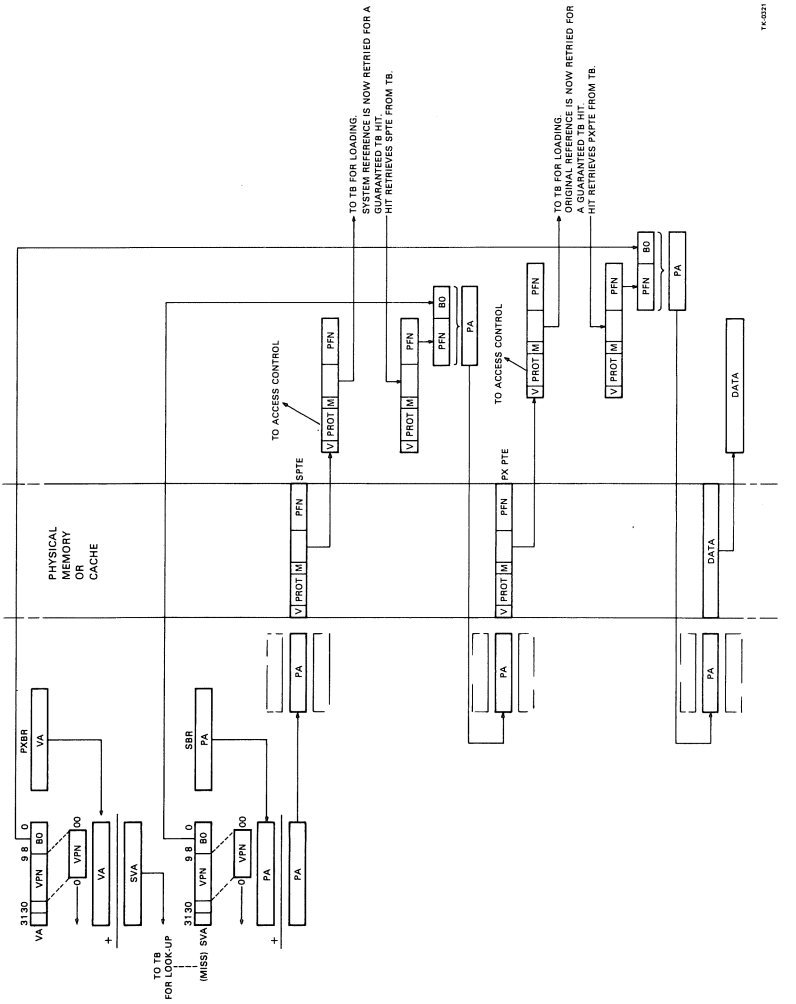
PROCESS VIRTUAL TO PHYSICAL ADDRESS TRANSLATION, EXAMPLE



ADDRESS CALCULATION FOR A TB HIT DURING A MISS MICROTRAP

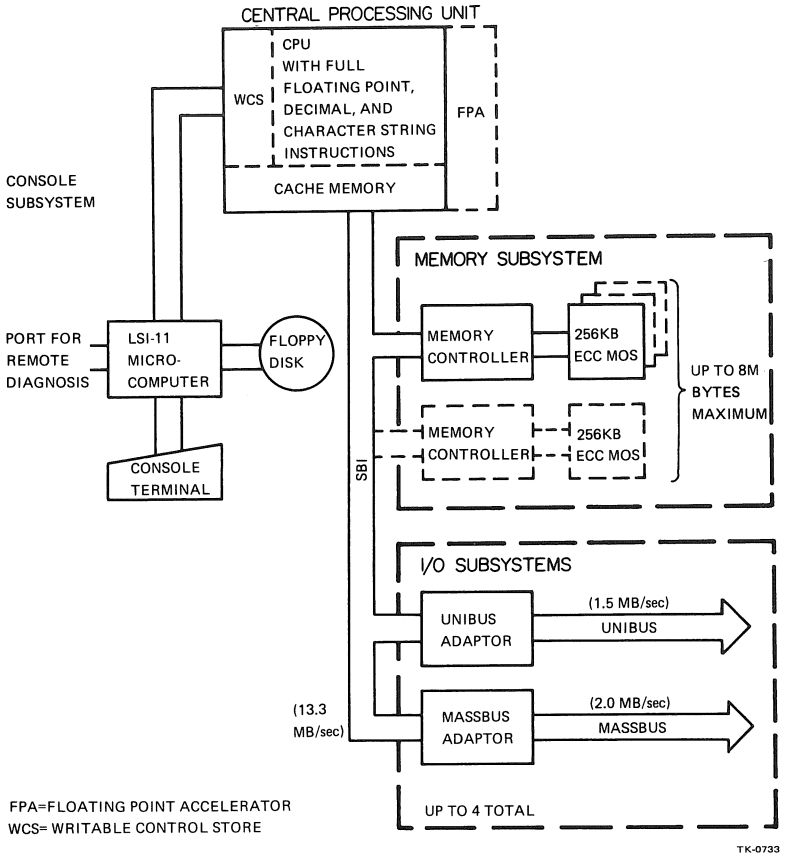


ADDRESS CALCULATION FOR A TB MISS DURING A MISS MICROTRAP

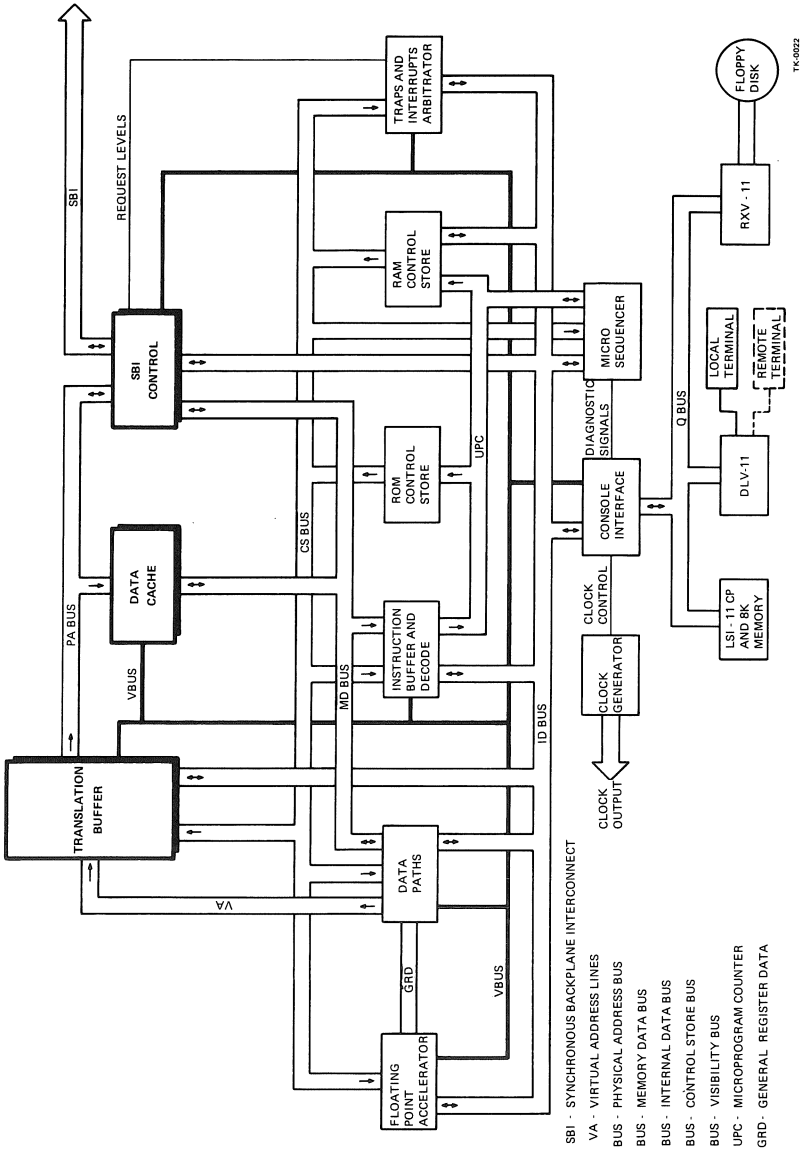


SECTION 3
HARDWARE BLOCK DIAGRAMS AND REGISTER BIT CONFIGURATIONS

VAX-11/780 GENERAL BLOCK DIAGRAM

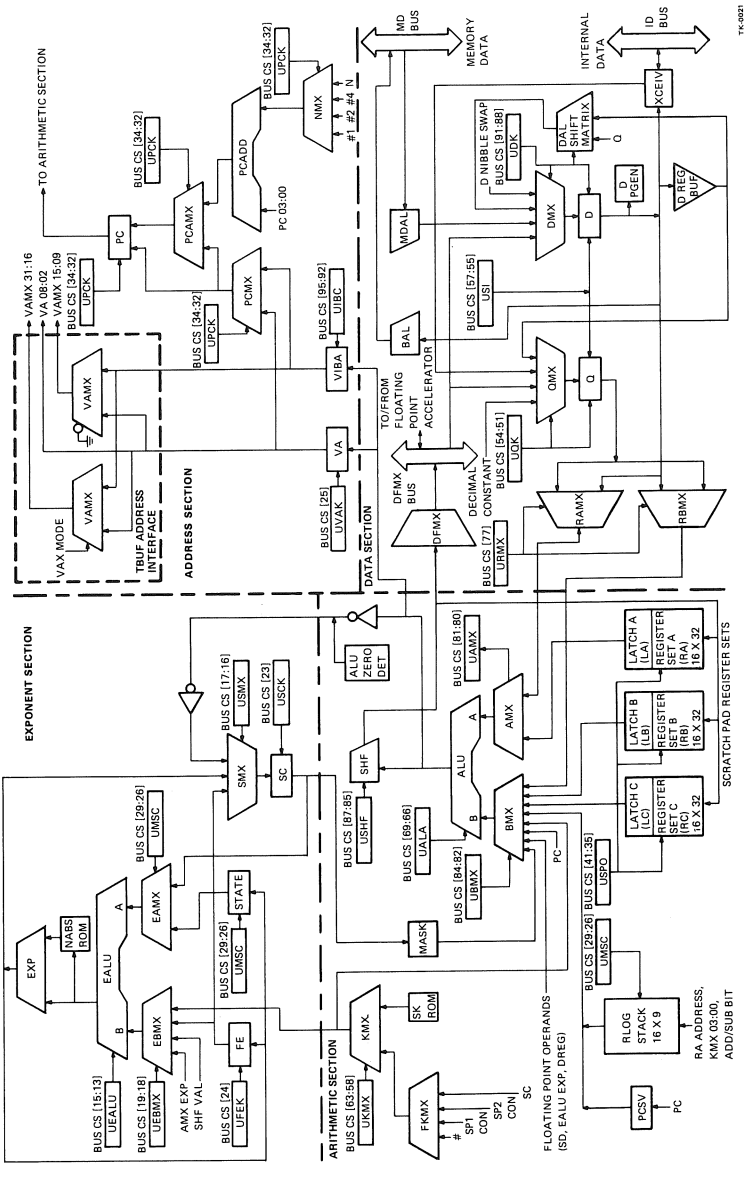


CPU BLOCK DIAGRAM

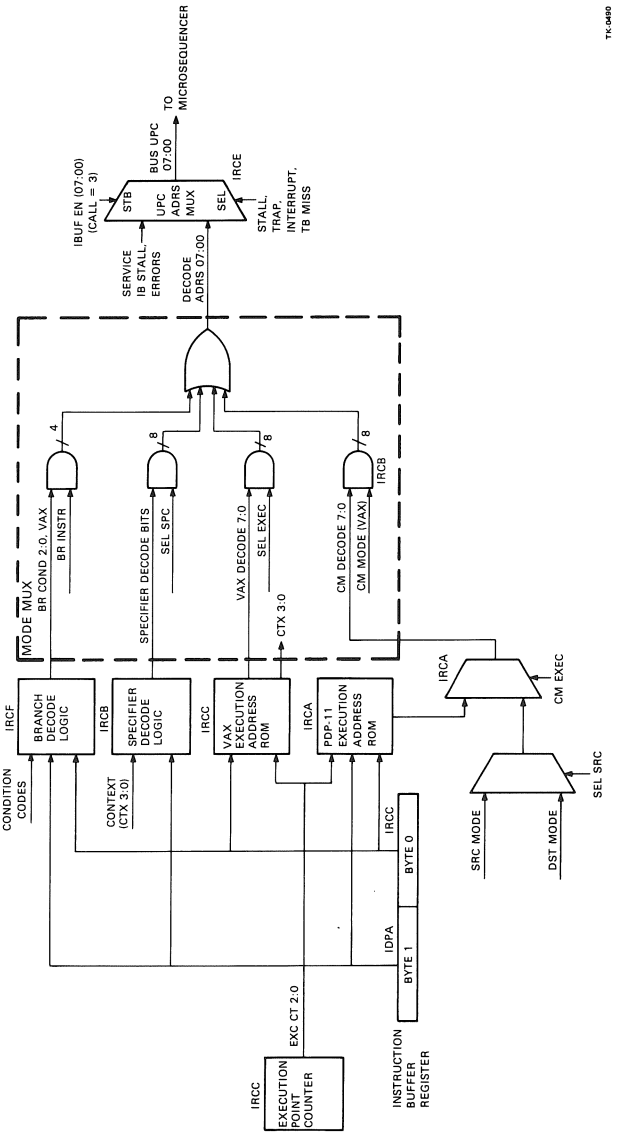


SBI - SYNCHRONOUS BACKPLANE INTERCONNECT
VA - VIRTUAL ADDRESS LINES
PA BUS - PHYSICAL ADDRESS BUS
MD BUS - MEMORY DATA BUS
ID BUS - INTERNAL DATA BUS
CS BUS - CONTROL STORE BUS
V BUS - VISIBILITY BUS
UPC - MICROPROGRAM COUNTER
GRD - GENERAL REGISTER DATA

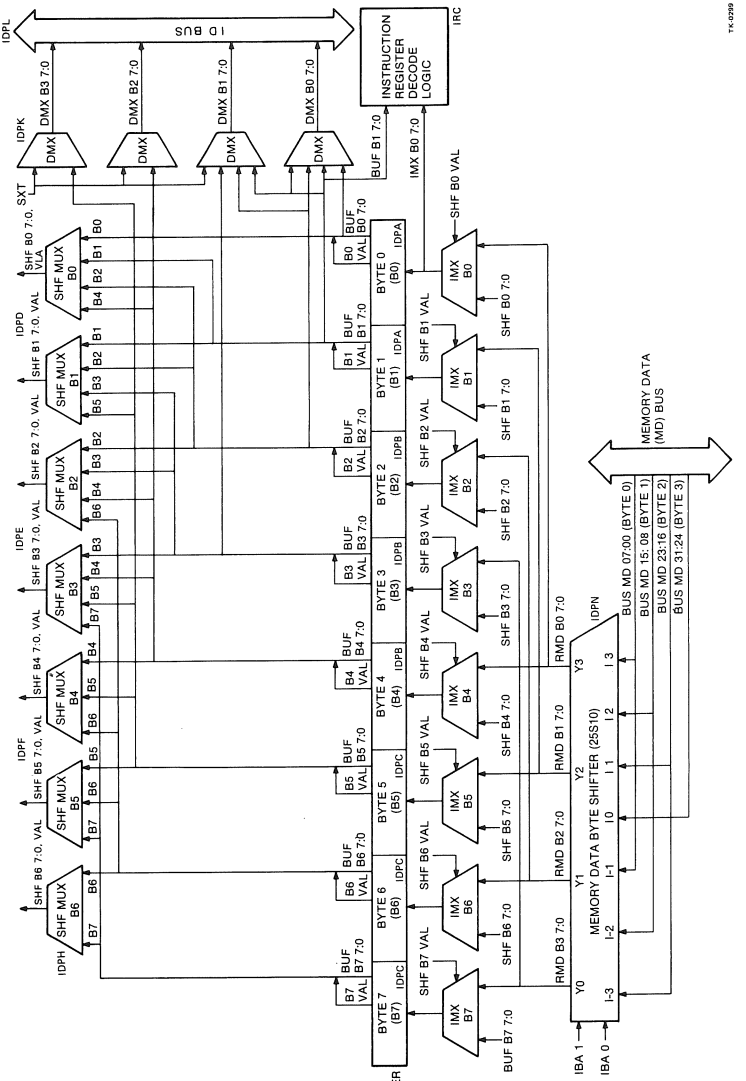
DATA PATH BLOCK DIAGRAM



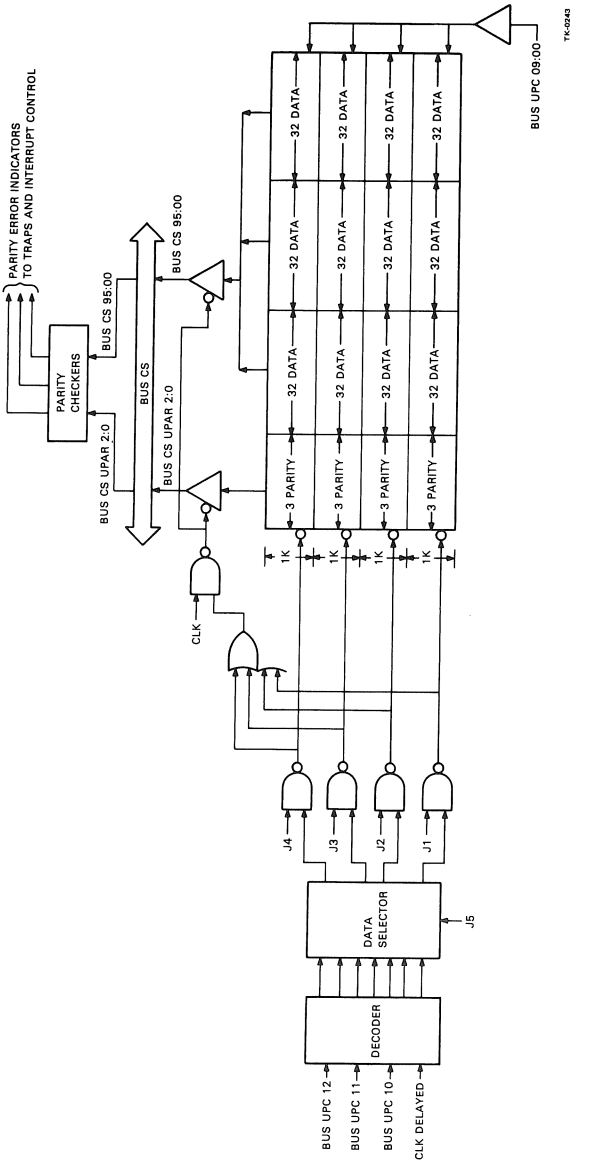
INSTRUCTION DECODE BLOCK DIAGRAM



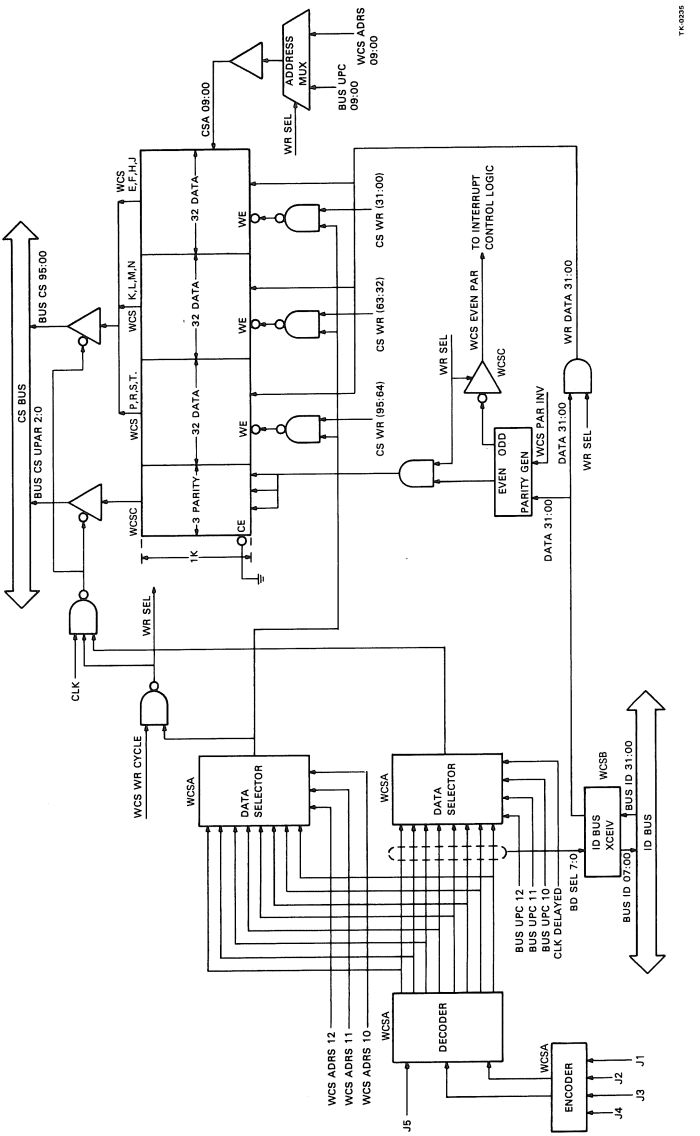
INSTRUCTION BUFFER BLOCK DIAGRAM



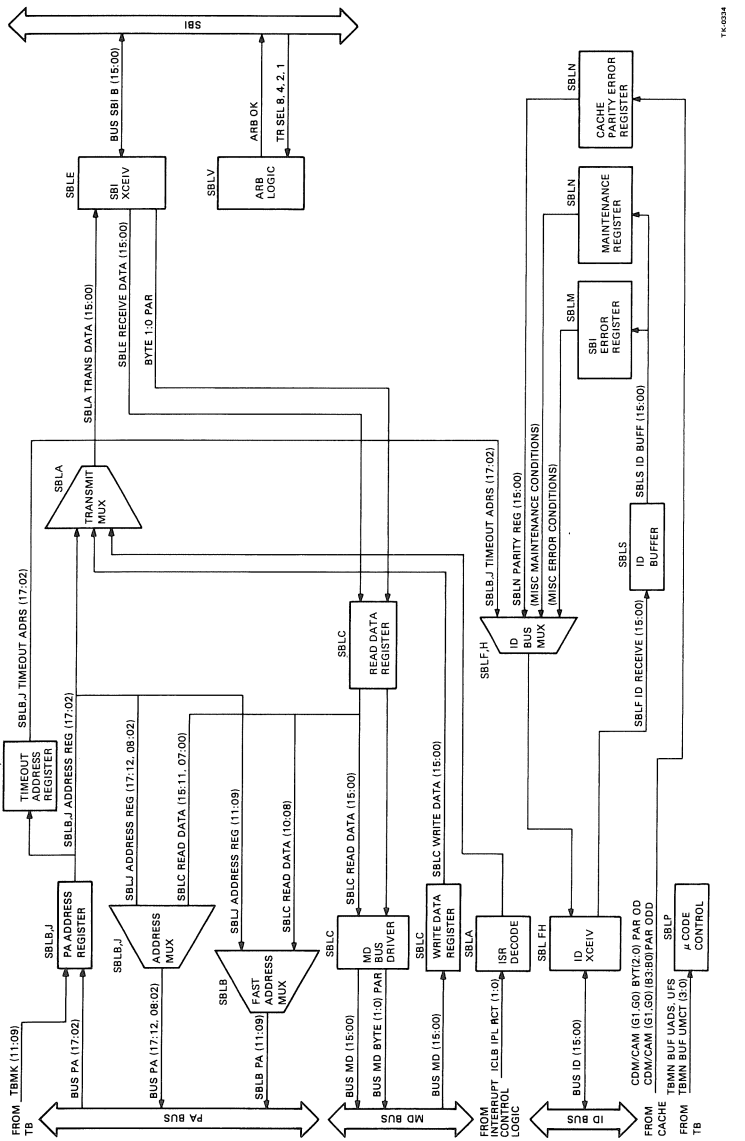
PROM CONTROL STORE (PCS) BLOCK DIAGRAM



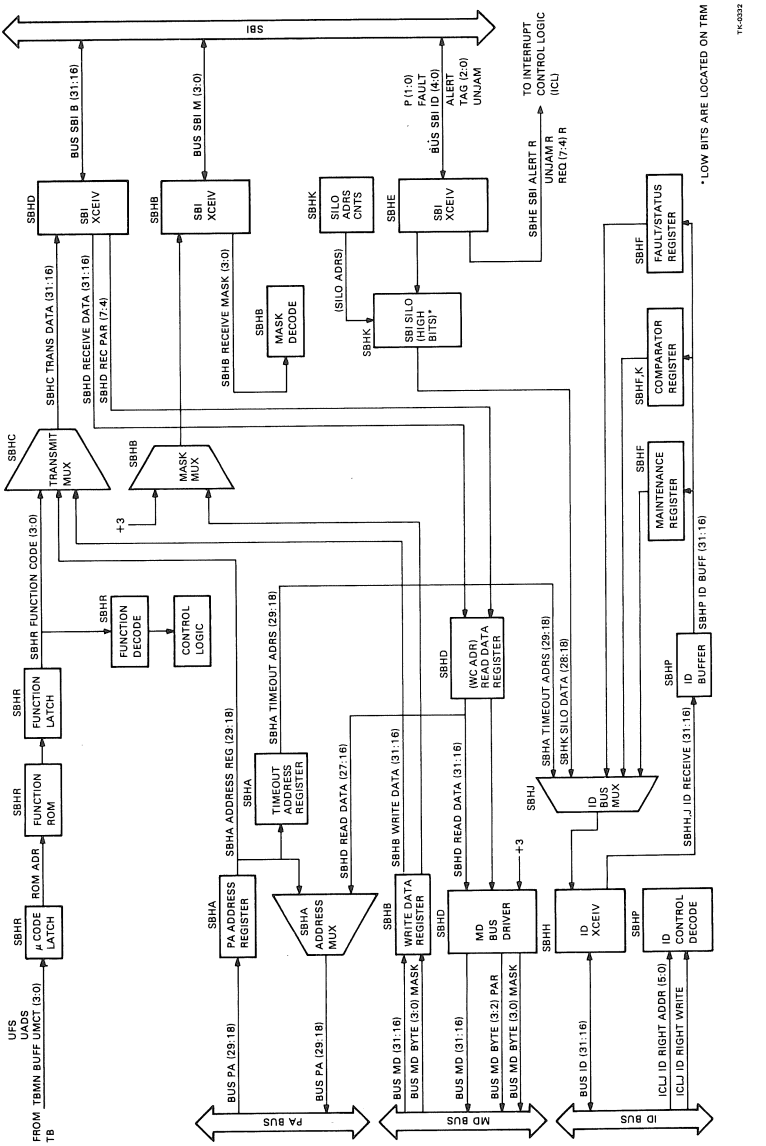
WRITABLE CONTROL STORE (WCS) BLOCK DIAGRAM



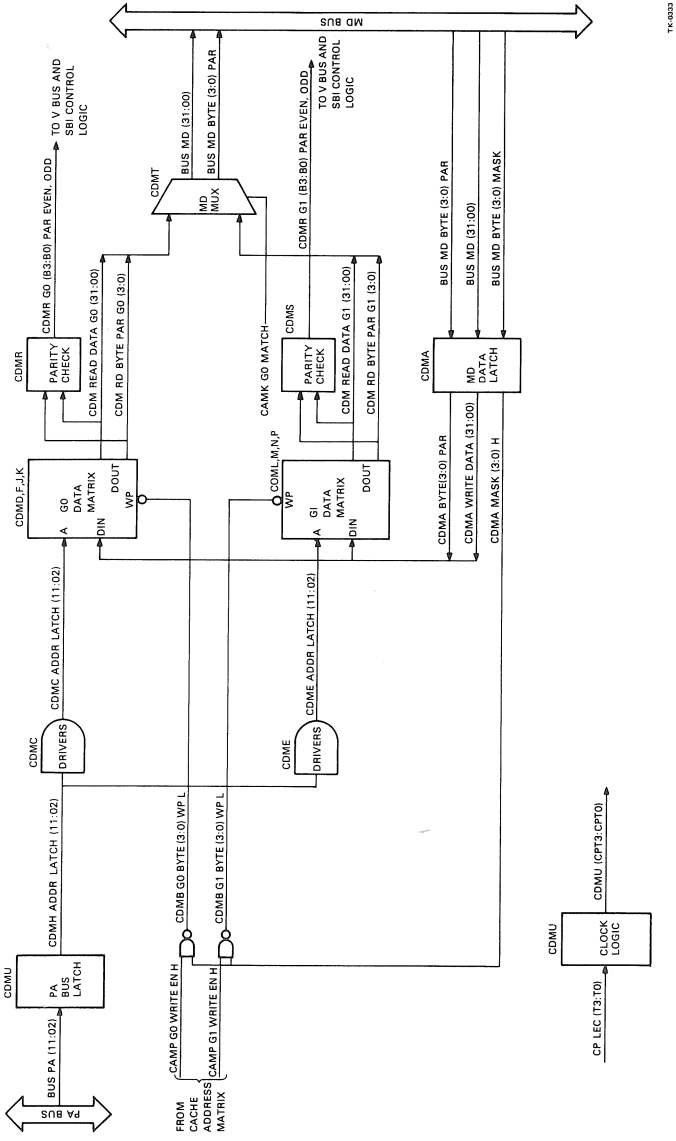
SBI CONTROL LOW BITS (SBL) BLOCK DIAGRAM



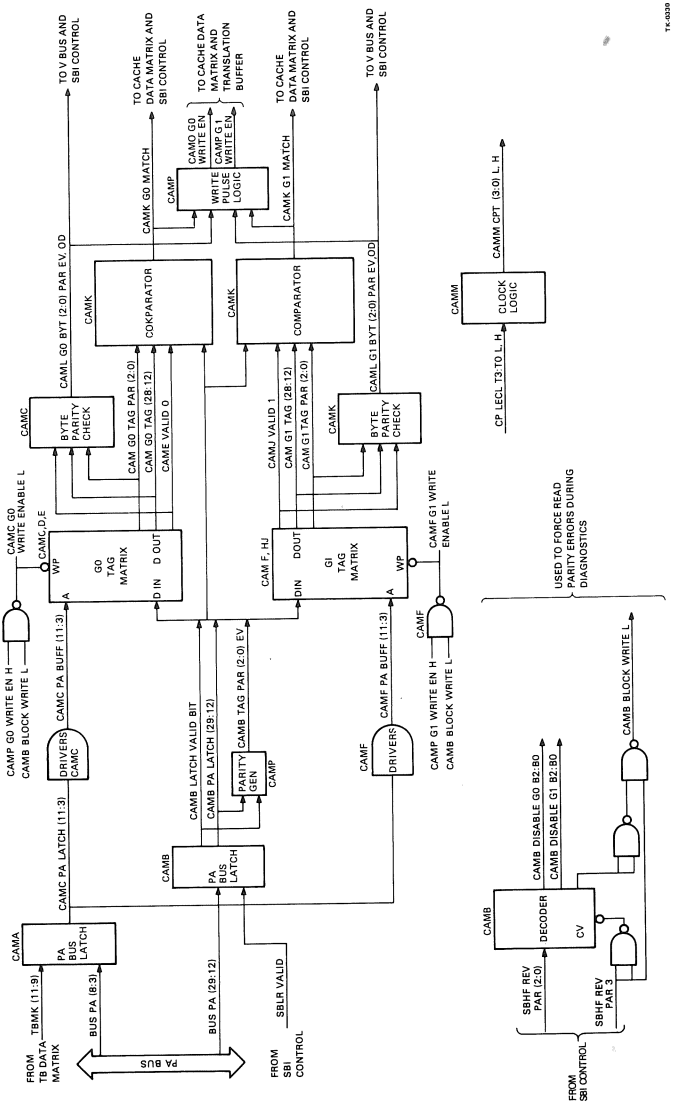
SBI CONTROL HIGH BITS (SBH) BLOCK DIAGRAM



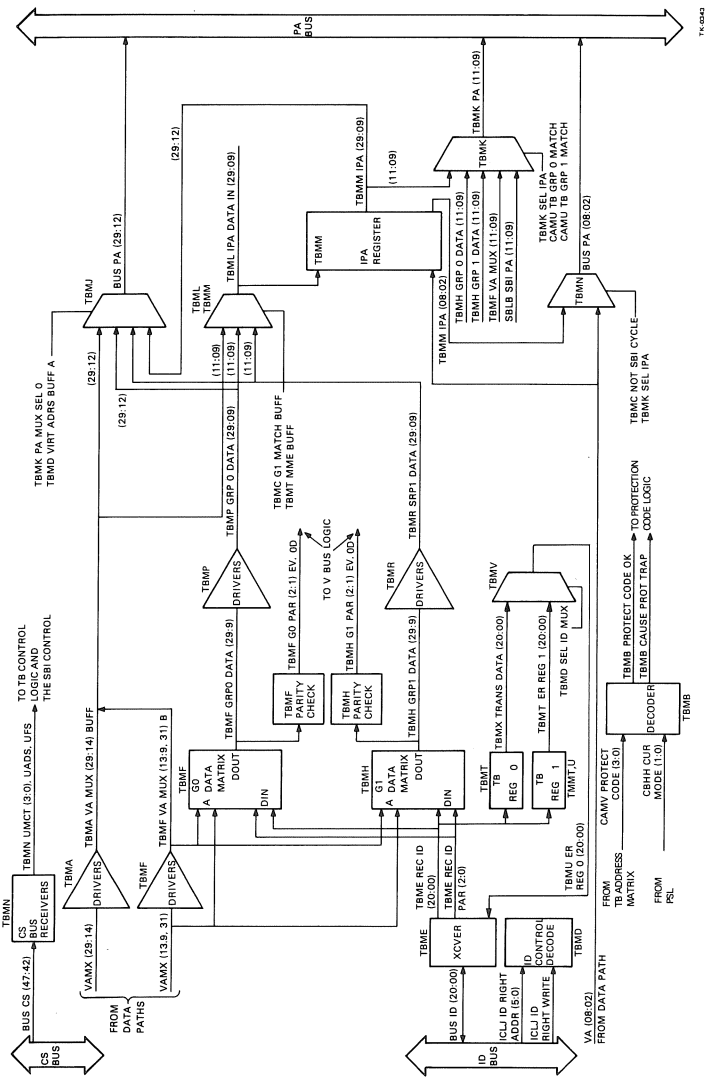
CACHE DATA MATRIX BLOCK DIAGRAM



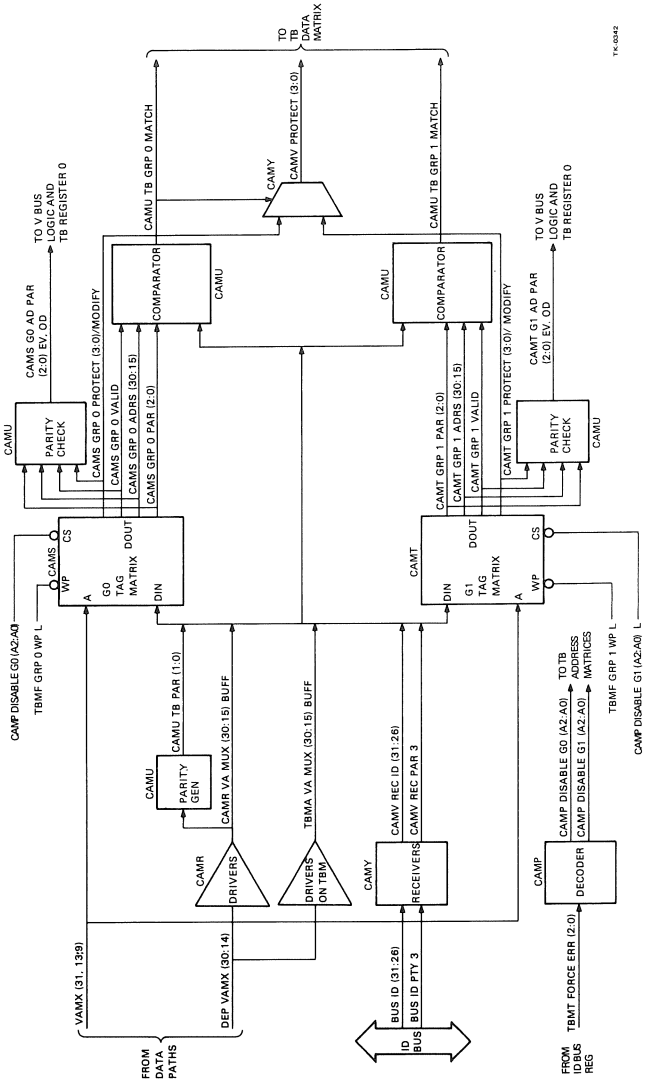
CACHE ADDRESS MATRIX BLOCK DIAGRAM



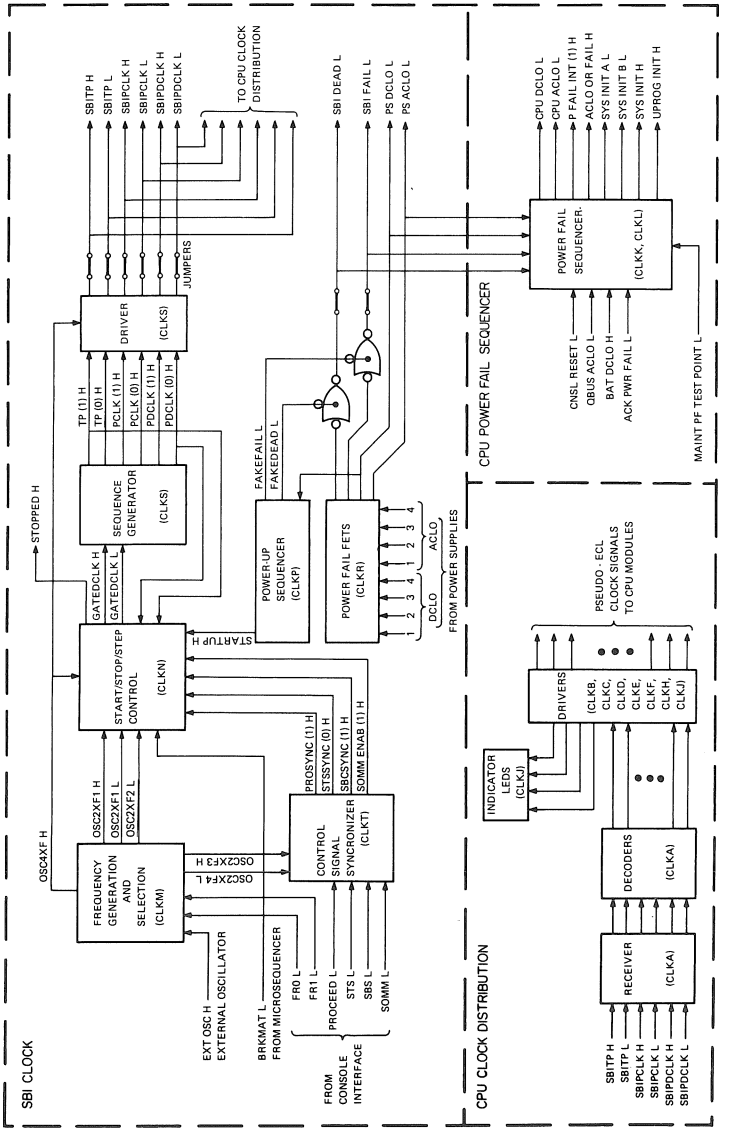
TRANSLATION BUFFER DATA MATRIX BLOCK DIAGRAM



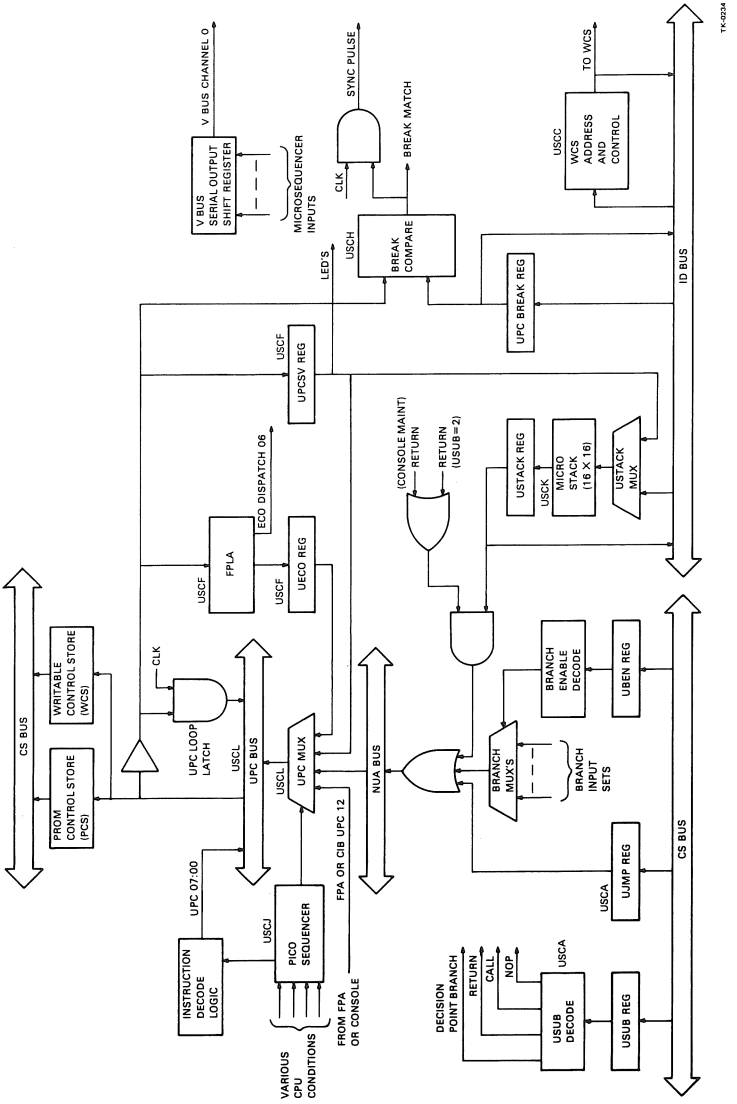
TRANSLATION BUFFER ADDRESS MATRIX BLOCK DIAGRAM

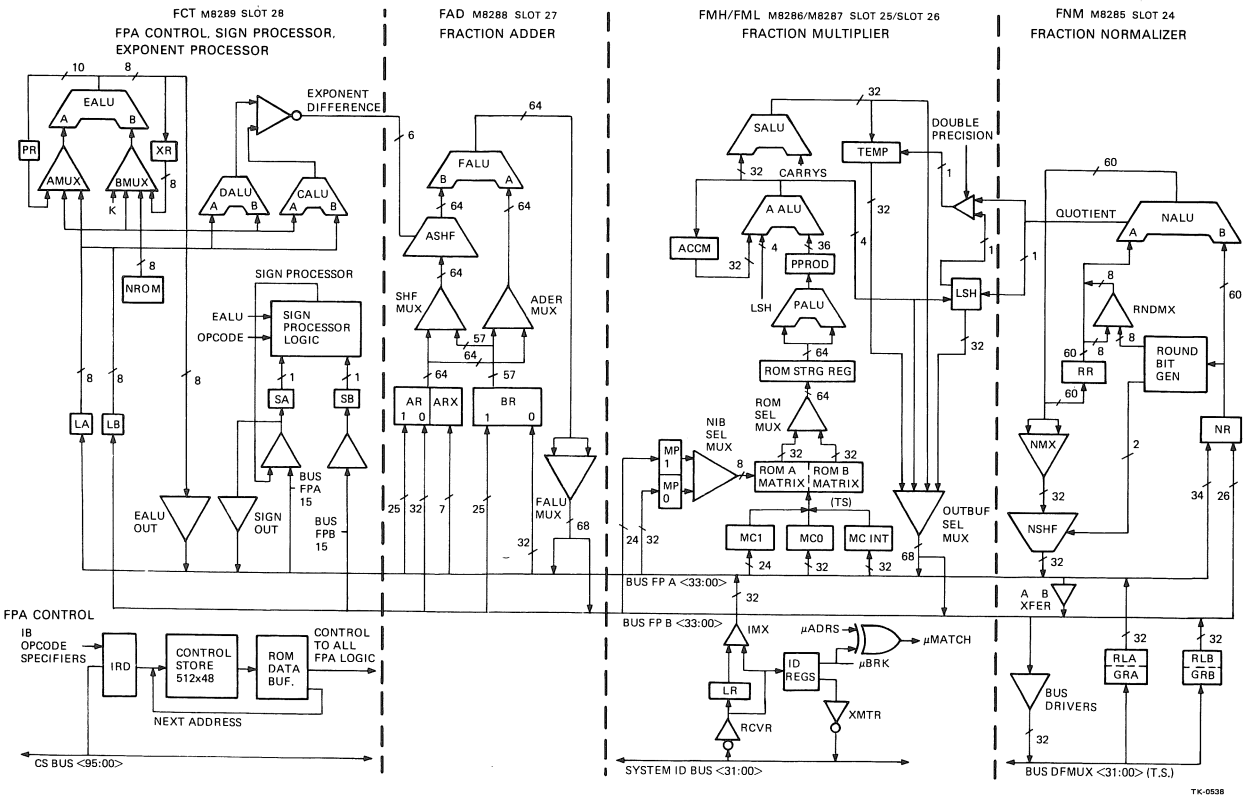


CLOCK MODULE BLOCK DIAGRAM

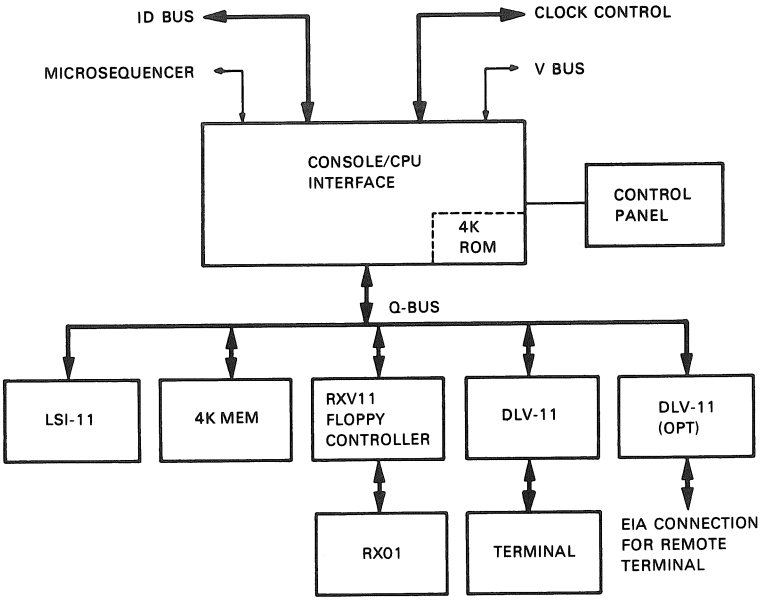


MICROSEQUENCER BLOCK DIAGRAM

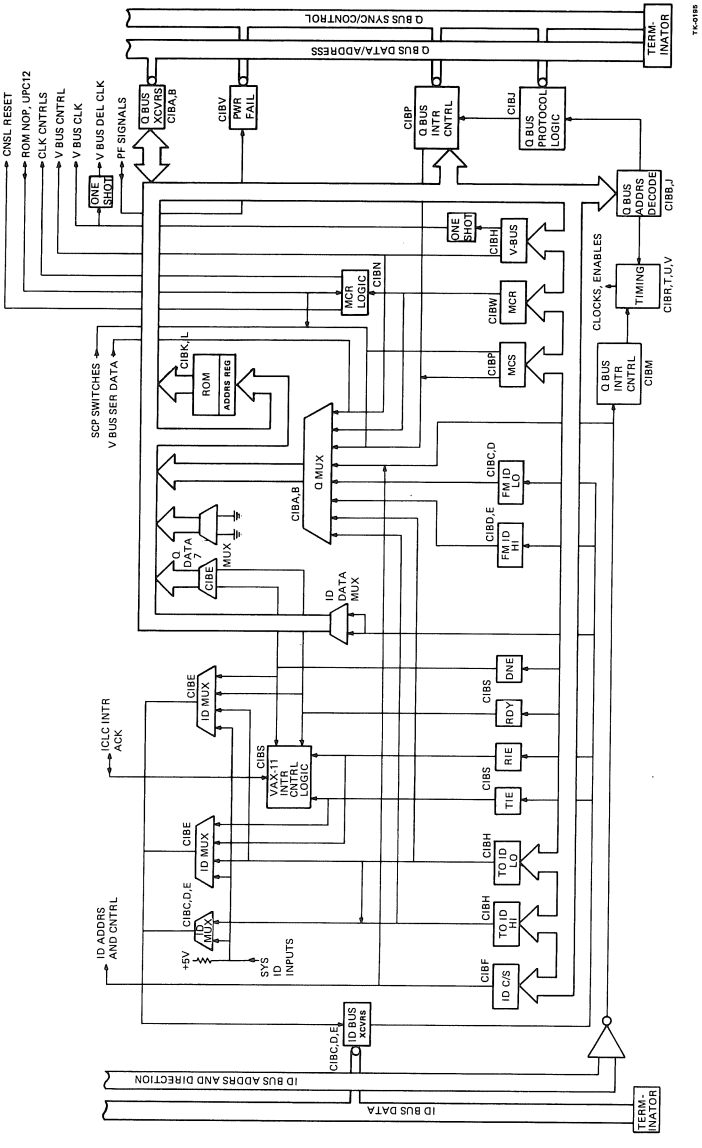



FCT M8289 SLOT 28
FPA CONTROL, SIGN PROCESSOR, EXPONENT PROCESSOR
FAD M8288 SLOT 27
FRACTION ADDER
FMH/FML M8286/M8287 SLOT 25/SLOT 26
FRACTION MULTIPLIER
FNM M8285 SLOT 24
FRACTION NORMALIZER
EALU
PR
XR
AMUX
BMUX
K
DALU
CALU
EXPONENT DIFFERENCE
NROM
SIGN PROCESSOR
EALU OPCODE
SIGN PROCESSOR LOGIC
SA
SB
LA
LB
BUS FPA 15
BUS FPB 15
EALU OUT
SIGN OUT
FALU
ASHF
SHF MUX
ADER MUX
AR
ARX
BR
FALU MUX
SALU
TEMP
DOUBLE PRECISION
CARRYS
A ALU
ACCM
PPROD
LSH
PALU
ROM STRG REG
ROM SEL MUX
NIB SEL MUX
MP 1
MP 0
ROM A MATRIX
ROM B MATRIX
(TS)
MC1
MC0
MC INT
OUTBUF SEL MUX
QUOTIENT
NALU
RNDMX
RR
ROUND BIT GEN
NR
NMX
NSHF
A B XFER
RLA GRA
RLB GRB
BUS DRIVERS
BUS FP A <33:00>
BUS FP B <33:00>
IMX
μADRS
μBRK
μMATCH
ID REGS
LR
RCVR
XMTR
SYSTEM ID BUS <31:00>
BUS DFMUX <31:00> (T.S.)
FPA CONTROL
IB OPCODE SPECIFIERS
IRD
CONTROL STORE 512x48
ROM DATA BUF.
CONTROL TO ALL FPA LOGIC
NEXT ADDRESS
CS BUS <95:00>
TK-0538

CONSOLE SUBSYSTEM CONFIGURATION



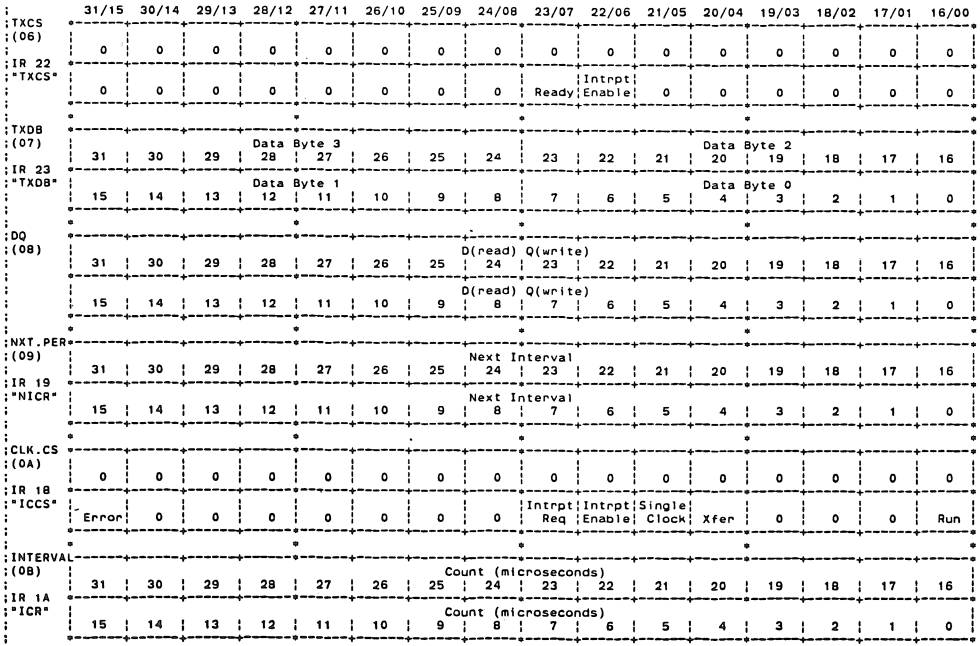
CONSOLE INTERFACE BOARD BLOCK DIAGRAM

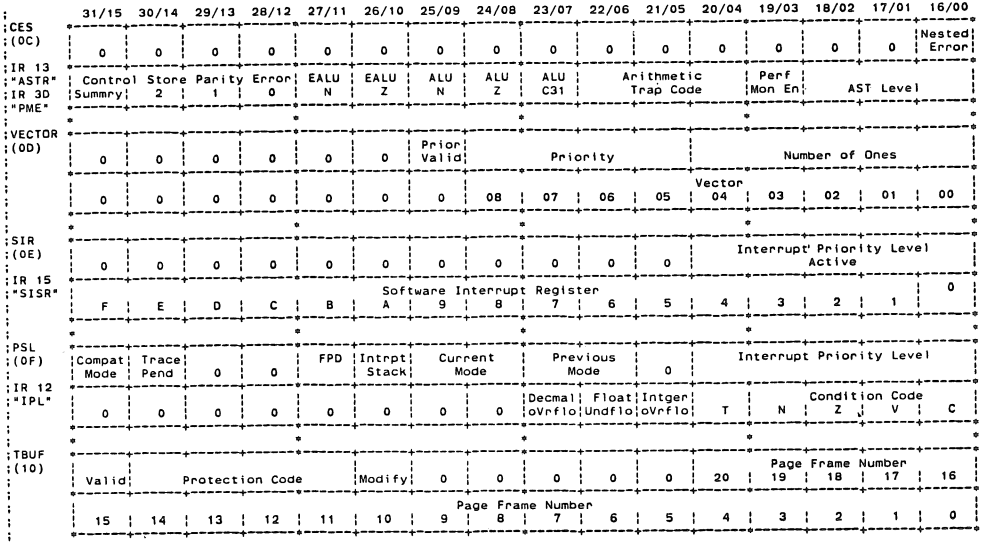


;September 2, 1977

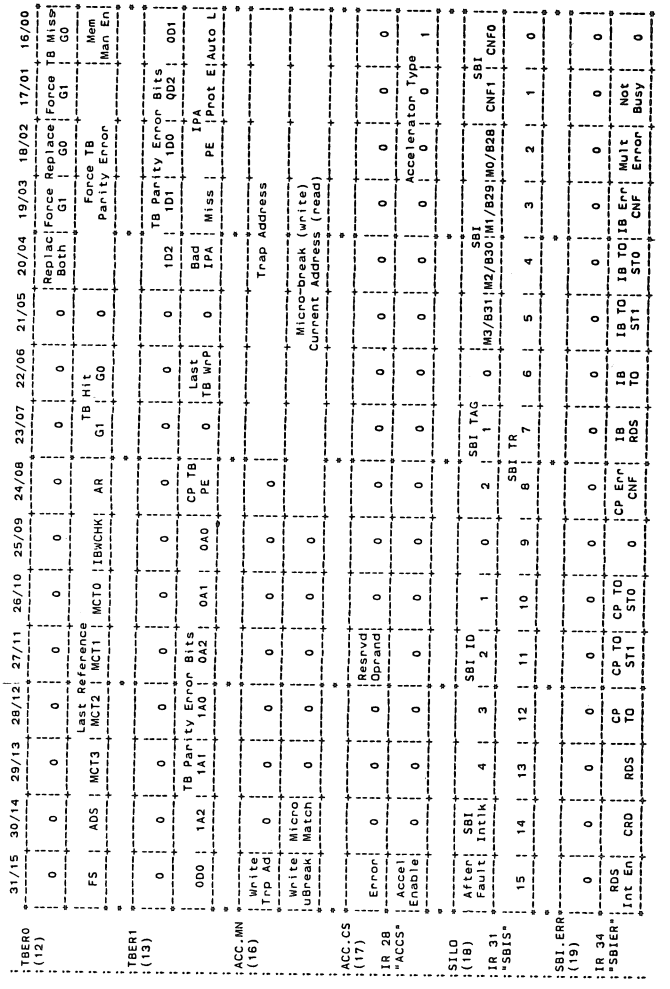
; The left column lists first the ID bus register name, and in parenthesis
; its address. If the register is also accessible by MTPR and MFPR inst-
; ructions, the Internal Register Number and symbolic name are also given.

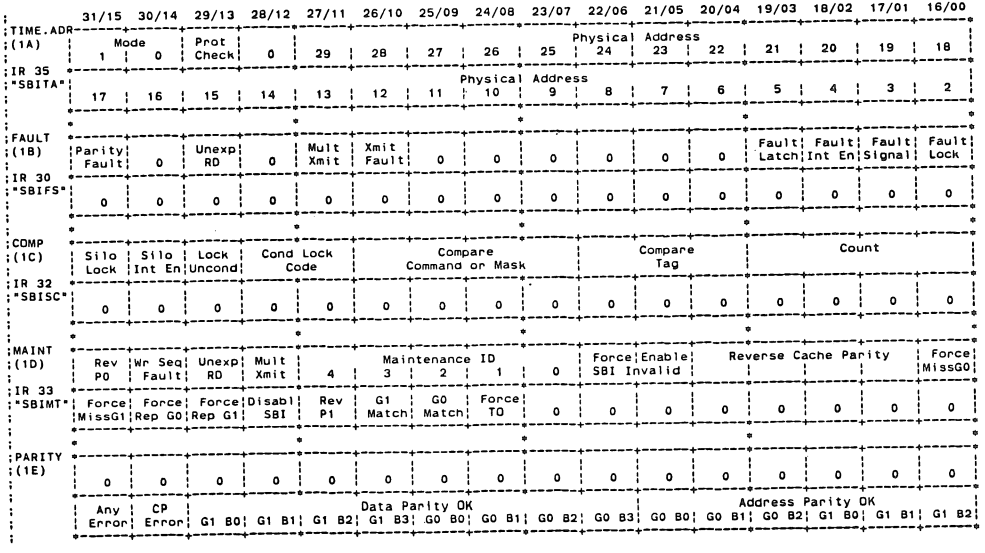
	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
IBUF (00)	Data Byte 3								Data Byte 2							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Data Byte 1								Data Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAY.TIM (01)	Time Byte 3								Time Byte 2							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IR 1B "TODR"	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)	ID 31	ID 30	ID 29	ID 28	ID 27	ID 26	ID 25	ID 24	ID 23	ID 22	ID 21	ID 20	ID 19	ID 18	ID 17	ID 16
IR 3E "SID"	ID 15	ID 14	ID 13	ID 12	ID 11	ID 10	ID 9	ID 8	ID 7	ID 6	ID 5	ID 4	ID 3	ID 2	ID 1	ID 0
RXCS (04)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IR 20 "RXCS"	0	0	0	0	0	0	0	0	Done	Intrpt Enable	0	0	0	0	0	0
RXDB (05)	Data Byte 3								Data Byte 2							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IR 21 "RXDB"	Data Byte 1								Data Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0


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)
IR 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
TXD8 (07)
IR 23 "TXD8"
Data Byte 3
Data Byte 2
Data Byte 1
Data Byte 0
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
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
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
NXT.PER (09)
IR 19 "NICR"
Next Interval
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
CLK.CS (0A)
IR 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)
IR 1A "ICR"
Count (microseconds)
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0


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)
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Nested Error
IR 13 "ASTR" IR 3D "PME"
Control Summry
Store 2
Parity 1
Error 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
Vector
0 0 0 0 0 0 0 08 07 06 05 04 03 02 01 00
SIR (0E)
0 0 0 0 0 0 0 0 0 0 0 0
Interrupt Priority Level Active
IR 15 "SISR"
Software Interrupt Register
F E D C B A 9 8 7 6 5 4 3 2 1 0
PSL (0F)
Compat Mode
Trace Pend
0 0
FPD
Intrpt Stack
Current Mode
Previous Mode
0
Interrupt Priority Level
IR 12 "IPL"
0 0 0 0 0 0 0 0
Decmal oVrflo
Float Undflo
Intger oVrflo
T
Condition Code
N Z V C
TBUF (10)
Valid
Protection Code
Modify
0 0 0 0 0
20 19 18 17 16
Page Frame Number
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

ID BUS MAP


TBER0 (12)
TBER1 (13)
ACC.MN (16)
ACC.CS (17)
IR 28 "ACCS"
SILO (18)
IR 31 "SBIS"
SBI.ERR (19)
IR 34 "SBIER"
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
FS ADS MCT3 MCT2 MCT1 MCT0 IBWCHK AR
Last Reference
TB Parity Error Bits
ODD 1A2 1A1 1A0 0A2 0A1 0A0
CP TB PE
TB Hit G1 G0
Last TB WrP
Write Trp Ad
Write uBreak
Micro Match
CRD RDS
CP ST1
CP TO ST0
CP Err CNF
IB RDS
IB TO
SBI TAG
SBI TR
SBI ID
Resrvd Operand
Error
Accel Enable
After Fault
Intlk
RDS Int En
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Micro-break (write)
Current Address (read)
Trap Address
Force TB Parity Error
TB Parity Error Bits
IPA
Replac Both
Force Replace G1 G0
Force TB G1 G0
TB Miss G0
Mem Man En
1D2 1D1 1D0 QD2 QD1
Bad IPA
Miss PE
Prot E Auto L
IB TO ST1
IB TO ST0
M3/B31 M2/B30 M1/B29 M0/B28
SBI
IB Err CNF
Mult Error
Not Busy
CNF1 CNF0
Accelerator Type
0 0 1



	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)	Mode 1	Mode 0	Prot Check	0	29	28	27	26	25	Physical Address 24	23	22	21	20	19	18
IR 35 "SBITA"	17	16	15	14	13	12	11	10	Physical Address 9	8	7	6	5	4	3	2
FAULT (1B)	Parity Fault	0	Unexp RD	0	Mult Xmit	Xmit Fault	0	0	0	0	0	0	Fault Latch	Fault Int En	Fault Signal	Fault Lock
IR 30 "SBIFS"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COMP (1C)	Silo Lock	Silo Int En	Lock Uncond	Cond Lock Code			Compare Command or Mask			Compare Tag			Count			
IR 32 "SBISC"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MAINT (1D)	Rev P0	Wr Seq Fault	Unexp RD	Mult Xmit	4	Maintenance ID 3	2	1	0	Force SBI Invalid	Enable		Reverse Cache Parity			Force MissGO
IR 33 "SBIMT"	Force MissG1	Force Rep G0	Force Rep G1	Disabl SBI	Rev P1	G1 Match	G0 Match	Force TO	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	Data Parity OK G1 B0	G1 B1	G1 B2	G1 B3	G0 B0	G0 B1	G0 B2	G0 B3	Address Parity OK G0 B0	G0 B1	G0 B2	G1 B0	G1 B1	G1 B2

	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
				12	11	10	9	8	Control Store Address 7	6	5	4	3	2	1	0
UBREAK (21)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IR 3C *MBRK*	0	0	0	12	11	10	9	8	Control Store Address 7	6	5	4	3	2	1	0
WCS.ADDR (22)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IR 2C *WCSA*	Invert Parity	Mod 3 Counter		12	11	10	9	8	Control Store Address 7	6	5	4	3	2	1	0
WCS.DATA (23)	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
IR 2D *WCSD*	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

Scratch Pad Locations

Name	Addr	IR #	Symb	Name	Addr	IR #	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	0B	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
<23:16>	ECO Level
<15:12>	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

ID BUS REGISTER BIT CONFIGURATIONS

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

ID BUS REGISTER BIT CONFIGURATIONS

	Located on: <7:00> M8228 (DCPC) <15:08> M8227 (DDPC) <31:16> M8226 (DEPL,M)
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> <8231 (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

ID BUS REGISTER BIT CONFIGURATIONS

	Allows counter to increment at 1 micro-second rate Read/Write Located on M8231 (ICLS)
ID#: 0B	NAME: INTERVAL COUNTER
Bit Fields	Description
<31:00>	32 Bit Up Counter At 1 micro-second 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

ID BUS REGISTER BIT CONFIGURATIONS

<08>	ALU Z
<07>	ALU C31 Read/Write Located on M8231 (ICLS)
<06:04>	Arithmetic Trap Code 7=Decimal divide by 0 6=Decimal Overflow 5=Float Underflow 4=Float divide by 0 3=Float Overflow 2=Integer divide by 0 1=Integer Overflow 0=No Trap Pending Read/Write Located on M8231 (ICLS)
<03>	Performance Monitor Enable Loaded or Read by Microcode Read/Write Located on M8231 (ICLS)
<02:01>	AST Level Used to deliver AST SIR during RET Read/Write Located on M8231 (ICLS)

ID#: 0D	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

ID BUS REGISTER BIT CONFIGURATIONS

	Located on M8230 (CEHP)
<20:16>	Number Of Ones
	Number of ones last written into vector register
	Read Only Located on M8230 (CEHP)
<08:00>	Vector
	Hardware Generated Vector
	Read Only Located on M8231 (ICLS)

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)
<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 Inst.) 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

ID BUS REGISTER BIT CONFIGURATIONS

<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>&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

ID BUS REGISTER BIT CONFIGURATIONS

	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)
<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

ID BUS REGISTER BIT CONFIGURATIONS

	<09> 1 means IB WCHK existed on an IB reference <08> 1 means Ref 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	NAME: TBUFF REG 1
Bit Fields	Description
<20:09>	TB Parity Error Status

ID BUS REGISTER BIT CONFIGURATIONS

	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 " " 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)
<08>	CP TB Parity Error Indicate TB Utrap has been requested Read/Any Write Clears Located on M8222 (TBME)
<06>	Last TB Write Pulse Indicates which TB group was last written 0=Group 0 1=Group 1 Both - Unpredictable Read Only Located on M8222 (TBME)
<04>	Bad IPA Contents of IPA are not meaningful Read Only Located on M8222 (TBME)
<03:00>	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 0=1 Automatic hardware initiated load Read Only Located on M8222 (TBME)
ID#: 16	NAME: ACCELERATOR MAINT.

ID BUS REGISTER BIT CONFIGURATIONS

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)
<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 reg will clear Located on M8286 (FMHR)
<27>	Reserved Operand Minus zero error

ID BUS REGISTER BIT CONFIGURATIONS

	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#: 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)

ID BUS REGISTER BIT CONFIGURATIONS

<17:16>	SBI CNF<1:0> Read Only Located on M8222 (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)
<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/Wrtie 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

ID BUS REGISTER BIT CONFIGURATIONS

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)

<03> 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)

<02> 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)

<01> SBI Not Busy

Read Only
Located on M8218 (SBLF)

ID BUS REGISTER BIT CONFIGURATIONS

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
<31:30>	Mode 31 30 0 0 Kernel 0 1 Executive 1 0 Supervisor 1 1 User Located on M8219 (SBHJ)
<29>	Protection Check Equal 0 for references not subject to hardware protection check Located on M8219 (SBHJ)
<27:00>	Physical Address <27:00>=PA<29:02> Located on <27:16> (SBHH,J) <16:00> (SBLF,H)
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)

ID BUS REGISTER BIT CONFIGURATIONS

<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)
<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

ID BUS REGISTER BIT CONFIGURATIONS

<31>	Comp Silo Lock A. Lock Unconditional (See bit 29) Locks when count (bits 19:16)=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 =F, silo will lock Unlock by writing number =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 =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)
<26:23>	Compare Command or Mask<3:0> Read/Write Located on M8219 (SBHJ) (SBHH)
<22:20>	Compare Tag<2:0> Read/Write Located on M8219 (SBHH)
<19:16>	Count Field<3:0>

ID BUS REGISTER BIT CONFIGURATIONS

	When =F allows silo lock Read/Write Located on M8219 (SBHH)
ID#: 1D	NAME: MAINTENANCE REGISTER
Bit Fields	Description
<31>	Force P0 Reversal on SBI Read/Write Located on M8219 (SBHJ)
<30>	Force Write Sequence Fault Read/Write Located on M8219 (SBHJ)
<29>	Force Unexpected Read Data Fault Causes transmit of SBI TAG=0, Maintenance ID, Undefined Data, good parity for unexpected read data in a selected nexus Read/Write Located on M8219 (SBHJ)
<27:23>	Maintenance ID<4:0> 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)
<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

ID BUS REGISTER BIT CONFIGURATIONS

```
                    20  19  18  17   Reverse Parity
                    0   0   0   0    No P
                    0   0   0   1    Group 1  Byte A  Address
                    0   0   1   0    Group 1  Byte B  Address
                    0   0   1   1    Group 1  Byte C  Address
                    0   1   0   0    Group 0  Byte A  Address
                    0   1   0   1    Group 0  Byte B  Address
                    0   1   1   0    Group 0  Byte C  Address
                    0   1   1   1    Unused
                    1   0   0   0    Group 1  Byte 3  Data
                    1   0   0   1    Group 1  Byte 2  Data
                    1   0   1   0    Group 1  Byte 1  Data
                    1   0   1   1    Group 1  Byte 0  Data
                    1   1   0   0    Group 0  Byte 3  Data
                    1   1   0   1    Group 0  Byte 2  Data
                    1   1   1   0    Group 0  Byte 1  Data
                    1   1   1   1    Group 0  Byte 0 Data

                    Read/Write
                    Located on M8219 (SBHH)

<16:15>     Force Cache Miss

                    16 15
                    0  0    No miss forced
                    0  1    Force miss Group 1
                    1  0    Force miss Group 0
                    1  1    Force miss Groups 0,1

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

<14:13>     Cache Replacement

                    14 13
                    0  0   Random
                    0  1   Group 1 always
                    1  0   Group 0 always
                    1  1   Undefined

                    Read/Write
                    Located on M8218 (SBLH)

<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>	Any Error Set when cache parity error on CP or IB read operations Read/Write 1 clears entire register Located on M8218 (SBLH)
<14>	CP Error When bits 15 and 14 are set, CP reference caused error When bit 15 is set and bit 14 is clear, the IB reference caused error Read Only Located on M8218 (SBLH)
<13:06>	Data Parity O.K. If Set Parity O.K., bit 15 must be set for meaningful info. 13 Parity OK CDM Group 1 Byte 0 12 " " " " 1 " 1 11 " " " " 1 " 2 10 " " " " 1 " 3 9 " " " " 0 " 0 8 " " " " 0 " 1 7 " " " " 0 " 2 6 " " " " 0 " 3

ID BUS REGISTER BIT CONFIGURATIONS

	Read Only Located on M8218 <13:8> (SBLH) <7:6> (SBLF)
<05:00>	Address Parity O.K. If Set Parity O.K., bit 15 must be set for meaningful info. 5 Parity OK CAM Group 0 Byte 0 4 " " " " 0 " 1 3 " " " " 0 " 2 2 " " " " 1 " 0 1 " " " " 1 " 1 0 " " " " 1 " 2 Read Only Located on M8218 (SBLF)
ID#: 20	NAME: USTACK
Bit Fields	Description
<15:00>	Reading pops top address from micro stack Writing pushes address on micro stack <15:00> = Control Store Address<15:00> Read/Write Located on M8235 (USCD)
ID#: 21	NAME: UBREAK
Bit Fields	Description
<12:00>	Data used to compare micro PC for scope sync or stopping system clock when SOMM set Read/Write Located on M8235 (USCD)
ID#: 22	NAME: WCS ADDRESS
Bit Fields	Description
<15>	Invert Parity When set inverts WCS parity Read/Write Located on M8235 (USCD)

ID BUS REGISTER BIT CONFIGURATIONS

<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

ID#	NAME	
24	P0BR	All Registers <31:00>
25	P1BR	
26	SBR	<31:24> M8230 CEHN
		<23:16> M8230 CEHM
28	KSP	<15:08> M8231 ICLR
29	ESP	<07:00> M8231 ICLP
2A	SSP	
2B	USP	
2C	ISP	ID Registers 24 then 2F
2D	FPDA	Are stored in A temps
2E	D.SY	on CEHK, ICLL
2F	Q.SY	
30	T0	30 thru 3E
31	T1	Are stored in B temps
32	T2	on CEHK, ICLL
33	T3	
34	T4	
35	T5	
36	T6	All registers
37	T7	Read/Write
38	T8	
39	T9	
3A	PCBB	
3B	SCBB	
3C	P0LR	
3D	P1LR	
3E	SLR	

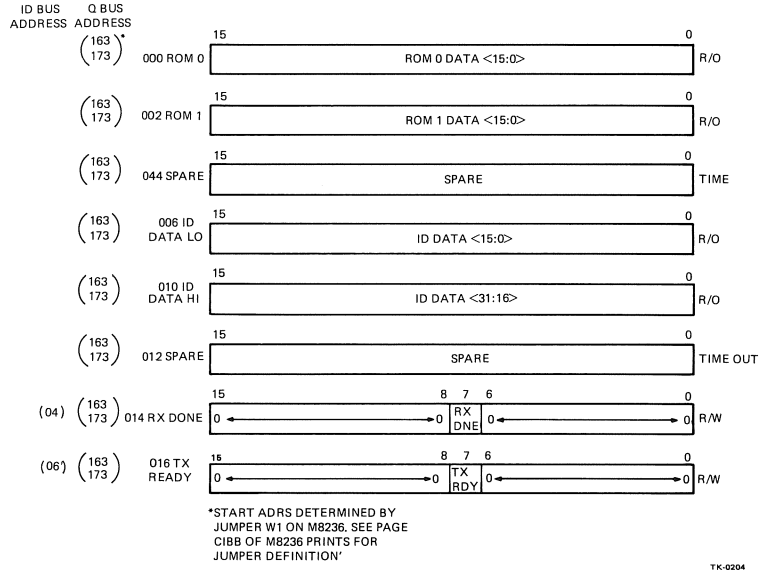
I/O Transfer Control Signals

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.

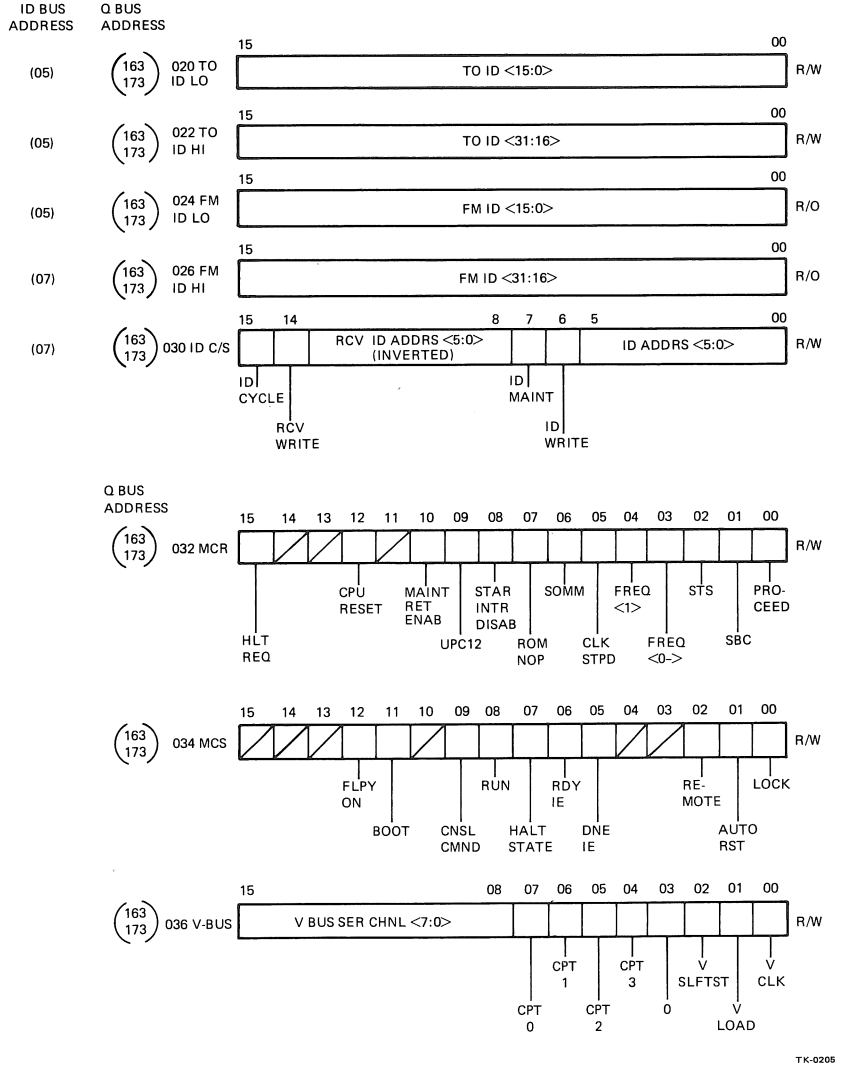
Name	Description
	I/O Transfer Control Signals
BWTBT L	Write/Byte – The LSI-11 processor uses BWTBT L to control bus cycles in two ways: 1. The processor asserts BWTBT L on the leading edge of BSYNC L to indicate that an output sequence (DATO or DATOB) is to follow. 2. The processor asserts BWTBT L together 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.

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.

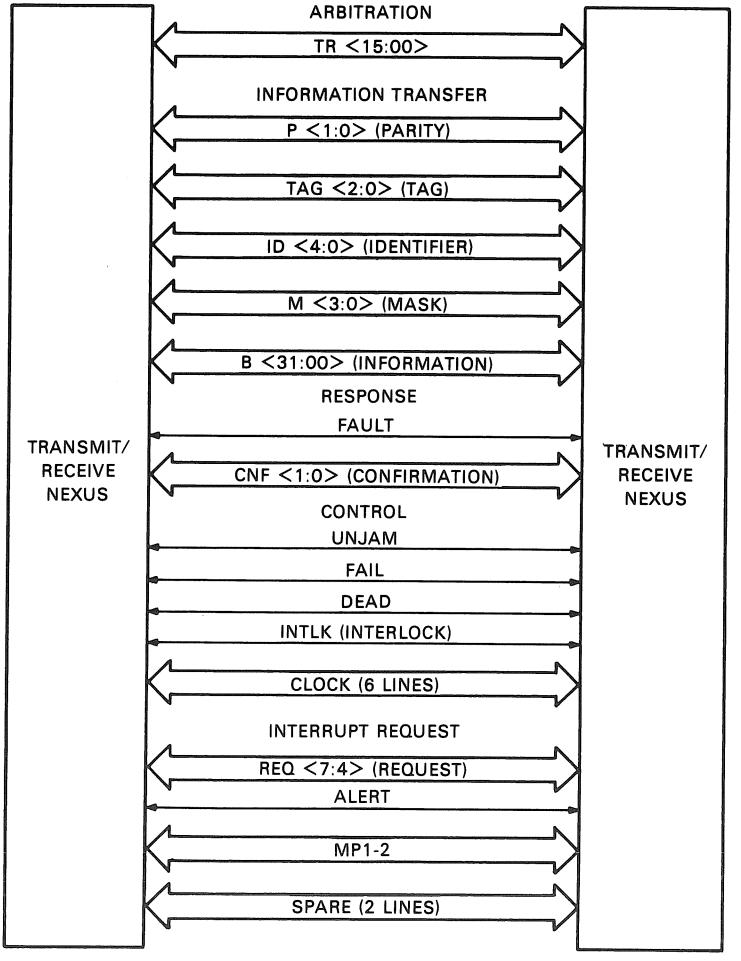
CIB Q BUS REGISTERS (LOWER)



CIB Q BUS REGISTERS (UPPER)

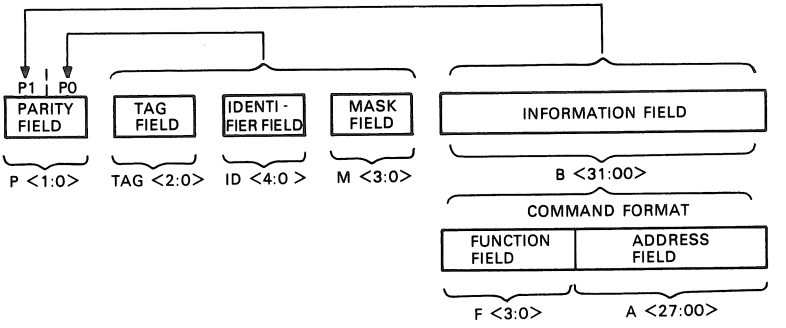


SBI CONFIGURATION



TK-0077

SBI PARITY FIELD CONFIGURATION



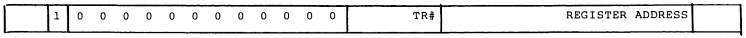
SBI FIELD DESCRIPTION

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

Field	Description
Interrupt Request Group	
Request Field [REQ (7:0)]	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.

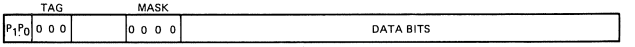
CONSOLE (PHYSICAL ADDRESS)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SBI ADDRESS			29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00



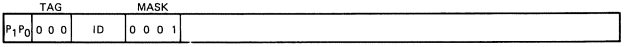
TR# (BASE 10)	PHYSICAL ADDRESS (HEX)	SBI ADDRESS (HEX)
0	20000000	8000000
1	20002000	8000800
2	20004000	8001000
3	20006000	8001800
4	20008000	8002000
5	2000A000	8002800
6	2000C000	8003000
7	2000E000	8003800
8	20010000	8004000
9	20012000	8004800
10	20014000	8005000
11	20016000	8005800
12	20018000	8006000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

SBI INFORMATION TRANSFER FORMATS

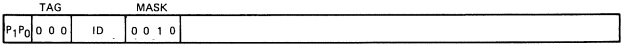
READ DATA FORMAT



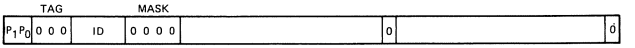
CORRECTED READ DATA FORMAT



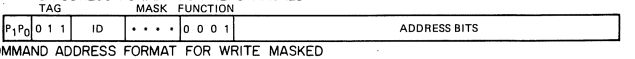
READ DATA SUBSTITUTE FORMAT



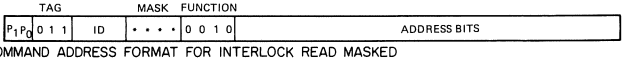
INTERRUPT SUMMARY RESPONSE FORMAT



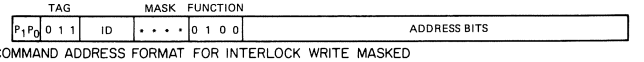
COMMAND ADDRESS FORMAT FOR READ MASKED



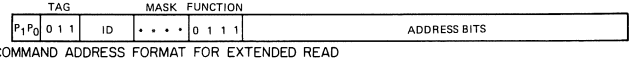
COMMAND ADDRESS FORMAT FOR WRITE MASKED



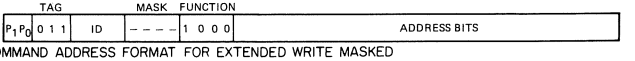
COMMAND ADDRESS FORMAT FOR INTERLOCK READ MASKED



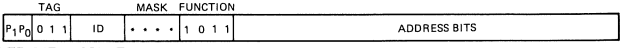
COMMAND ADDRESS FORMAT FOR INTERLOCK WRITE MASKED



COMMAND ADDRESS FORMAT FOR EXTENDED READ

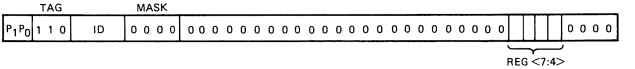


COMMAND ADDRESS FORMAT FOR EXTENDED WRITE MASKED



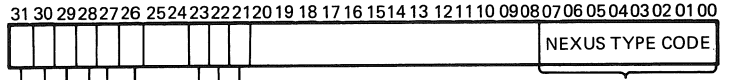
WRITE DATA FORMAT

INTERRUPT SUMMARY READ FORMAT



TK-0723

SBI FAULTS



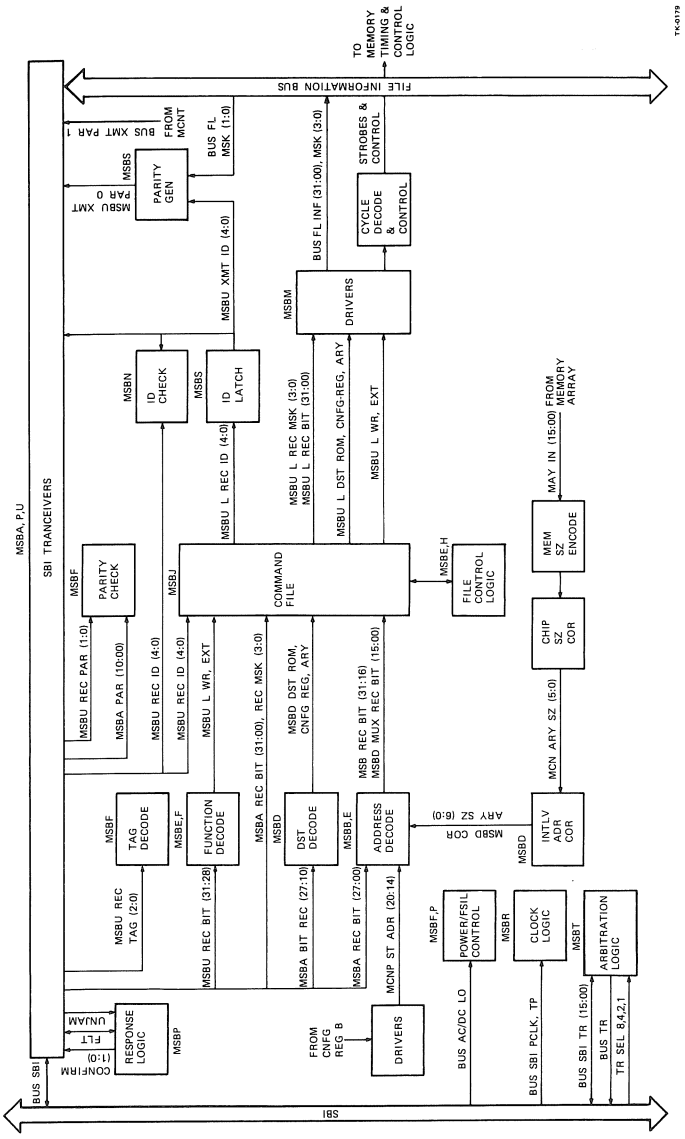
31	PARITY FAULT
30	WRITE SEQUENCE FAULT
29	UNEXPECTED READ DATA FAULT
28	INTERLOCK SEQUENCE FAULT
27	MULTIPLE TRANSMITTER FAULT
26	TRANSMITTER CURING CYCLE THAT CAUSED FAULT
23	POWER DOWN
22	POWER UP
21	OVER TEMPERATURE

000XXXXX	MEMORY	
0010000 0	MBA	0
0010000 1		1
0010001 0		2
0010001 1		3
0010010 0		4
0010010 1		5
0010011 0		6
0010011 1		7
0010100 0	UBA	0
0010100 1		1
0010101 0		2
0010101 1		3

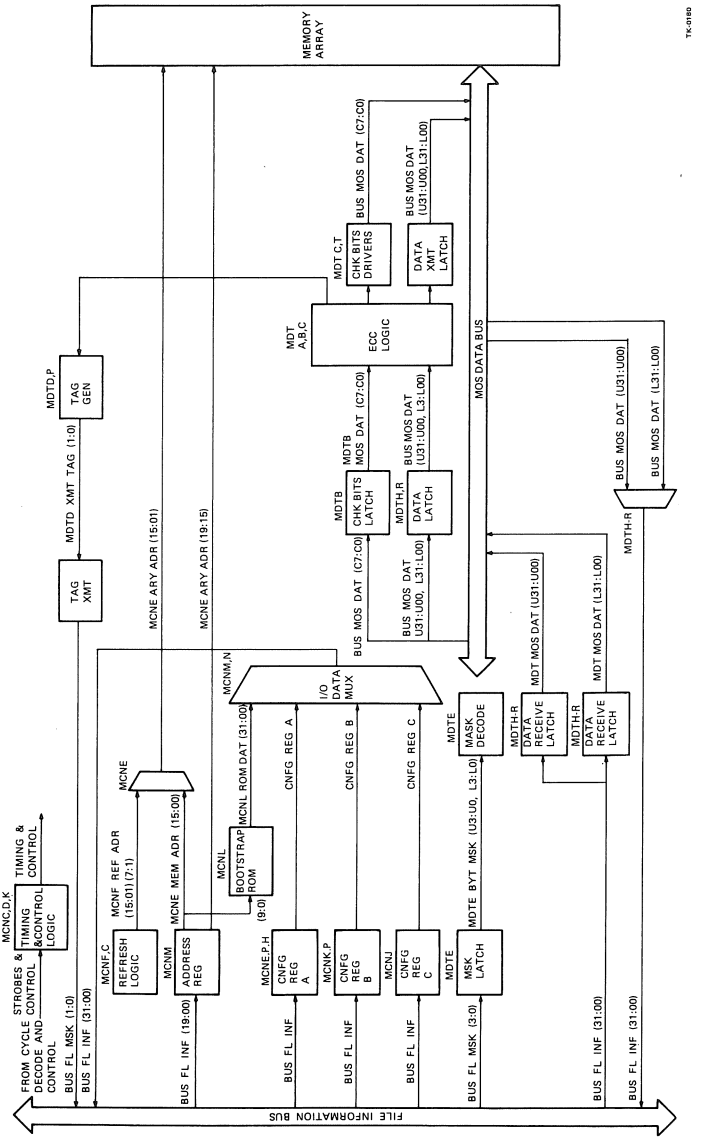
TK-0695

	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			

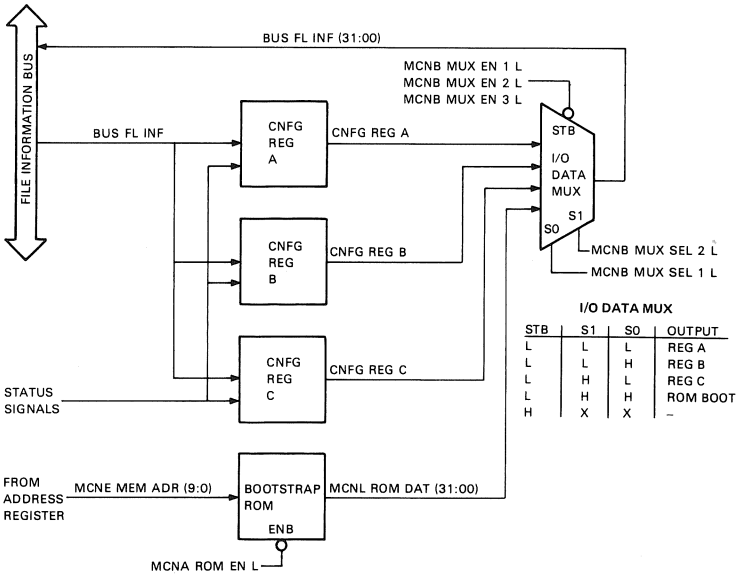
MEMORY BLOCK DIAGRAM, PART 1



MEMORY BLOCK DIAGRAM, PART 2



MEMORY I/O DATA LOGIC



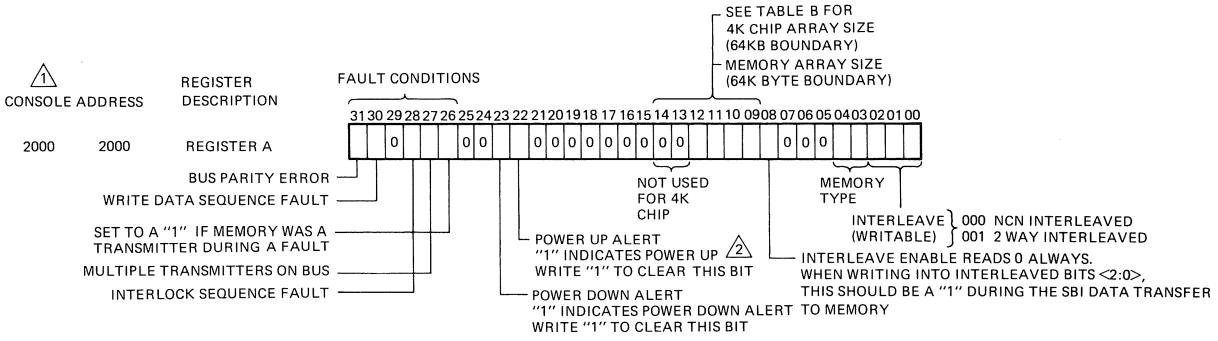


TABLE A 4K CHIP

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

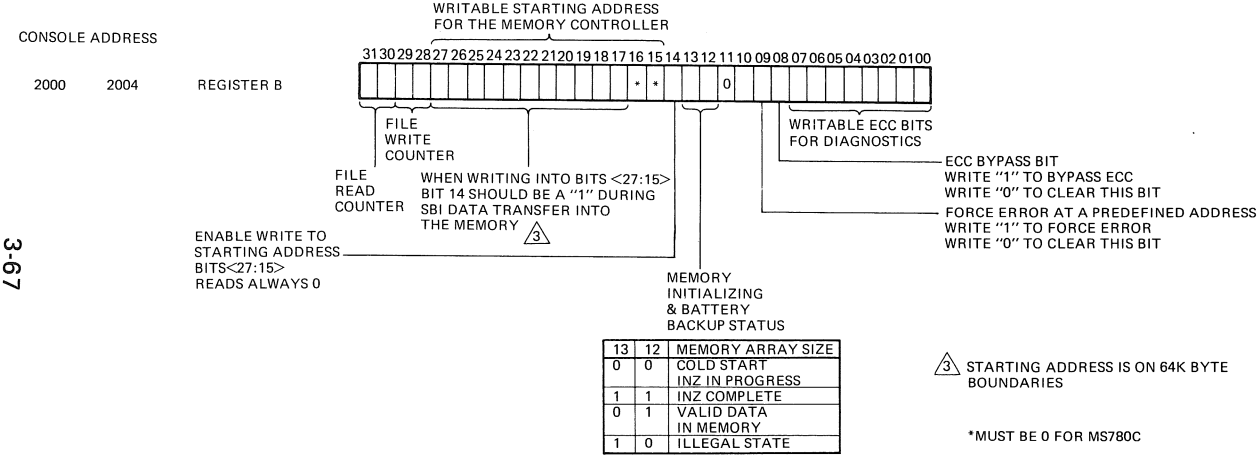
TABLE B 16K CHIP

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 WITHOUT ROM BOOTSTRAP

TK-0777C

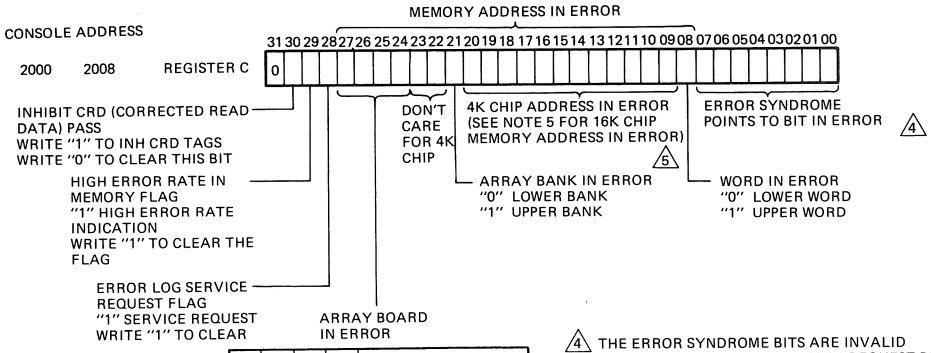


13	12	MEMORY ARRAY SIZE
0	0	COLD START INZ IN PROGRESS
1	1	INZ COMPLETE
0	1	VALID DATA IN MEMORY
1	0	ILLEGAL STATE

3 STARTING ADDRESS IS ON 64K BYTE BOUNDARIES

*MUST BE 0 FOR MS780C

TK-0777B



27	26	25	24	MEMORY ARRAY SIZE
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

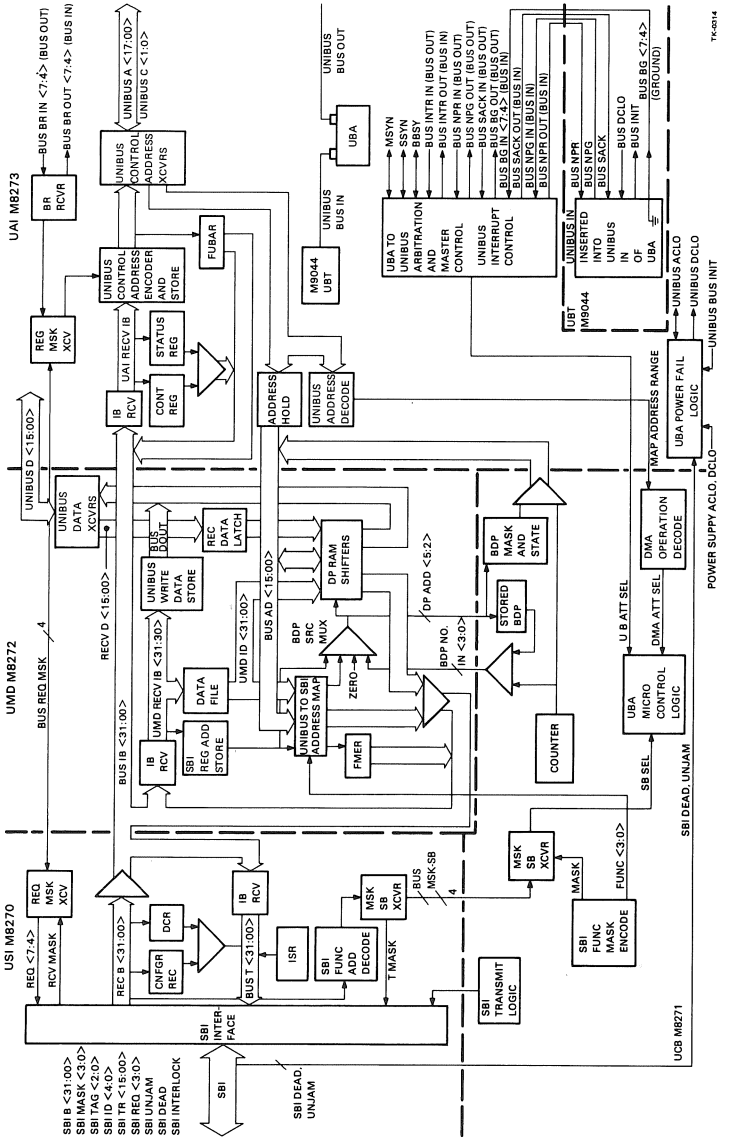
4 THE ERROR SYNDROME BITS ARE INVALID UNLESS THE ERROR LOG SERVICE REQUEST BIT (BIT 28, REG 0) IS SET TO A "1".

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 MS780A & MS780C ON THE SAME 11780, THE MS780C MUST HAVE THE LOWEST TR.

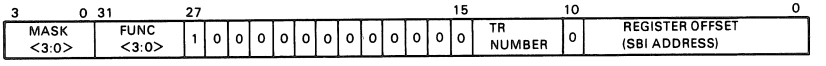
TK-0777A

UBA (DW780) BLOCK DIAGRAM



UBA ADDRESS SPACE AND C/A FORMAT

SBI C/A Format for UBA Register Access



TK-0320

Base Addresses

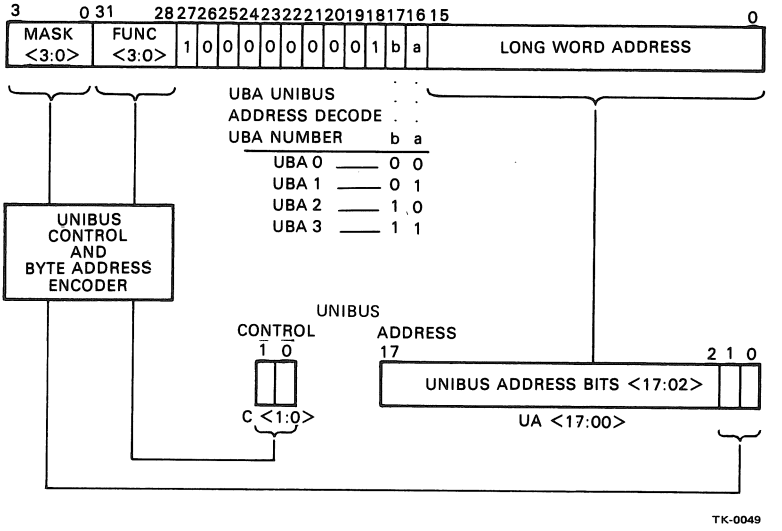
TR Num Base 10	Base Address (Physical hex)	SBI Address (hex)	TR Num Base 10	Base Address (Physical hex)	SBI Address (hex)
			8	20010000	8004000
1	20002000	8000800	9	20012000	8004800
2	20004000	8001000	10	20014000	8005000
3	20006000	8001800	11	20016000	8005800
4	20008000	8002000	12	20018000	8006000
5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

Register Offsets

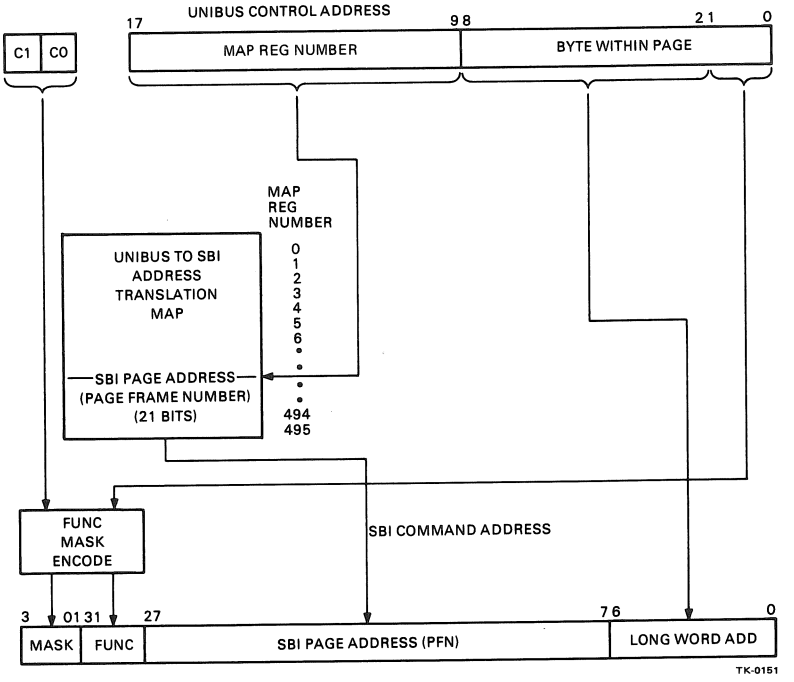
UBA Reg	Byte Address (Physical hex)	SBI Address (hex)	UBA Reg	Byte Address (Physical hex)	SBI Address (hex)
CNFGR	000	000	.	.	.
UBACR	004	001	.	.	.
UBASR	008	002	DPR 14	078	01E
DCR	00C	003	DPR 15	07C	01F
FMER	010	004	Reserved	080	020
FUBAR	014	005	.	.	.
FMER	018	006	.	.	.
FUBAR	01C	007	Reserved	7EC	1FF
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	EB8	3EE
BRRVR 5	034	00D	MR 495	EBC	3EF
BRRVR 6	038	00E	Reserved	EC0	3F0
BRRVR 7	03C	00F	.	.	.
DPR 0	040	010	Reserved	EFC	3FF
DPR 1	044	011			

SBI TO UNIBUS CONTROL ADDRESS TRANSLATION

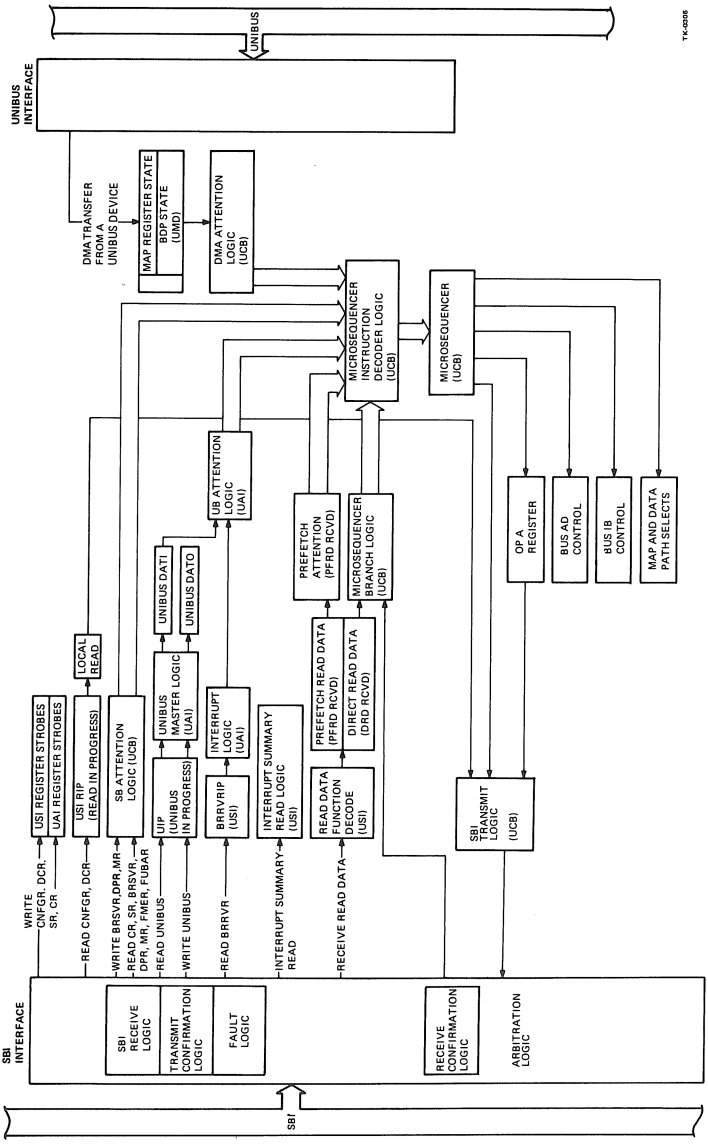
SBI COMMAND ADDRESS FORMAT



UNIBUS TO SBI ADDRESS TRANSLATION

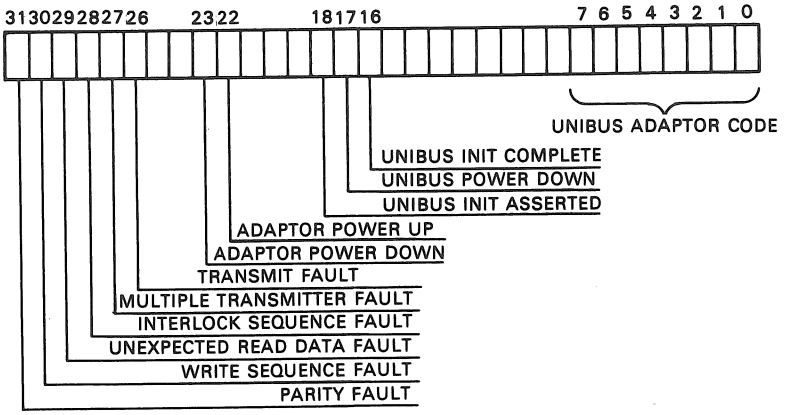


SIMPLIFIED FLOW OF MAJOR CONTROL FUNCTIONS WITHIN THE UBA



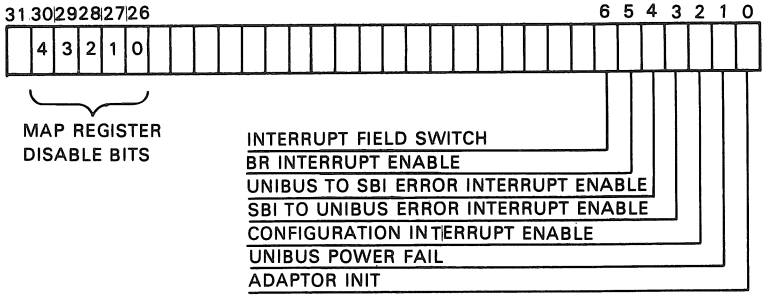
UBA REGISTERS

UBA CONFIGURATION REGISTER, BIT CONFIGURATION



TK-0119

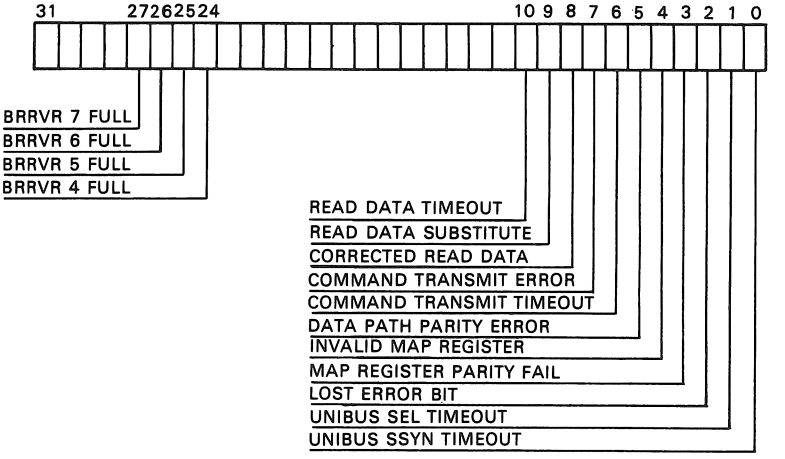
UBA CONTROL REGISTER, BIT CONFIGURATION



TK-0120

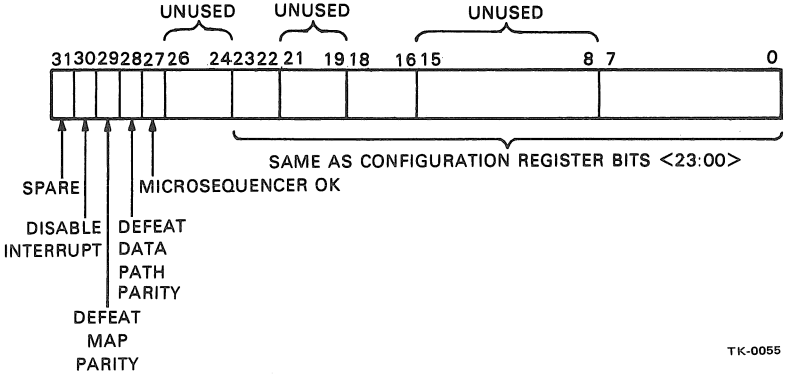
UBA REGISTERS

UBA STATUS REGISTER, BIT CONFIGURATION



TK-0121

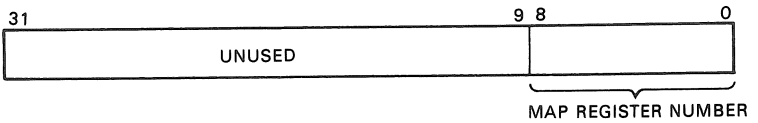
UBA DIAGNOSTIC CONTROL REGISTER, BIT CONFIGURATION



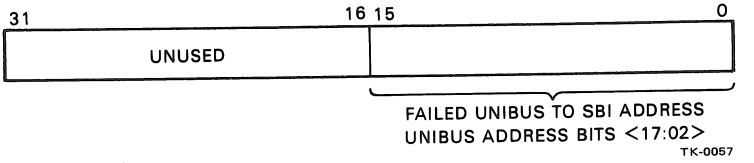
TK-0055

UBA REGISTERS

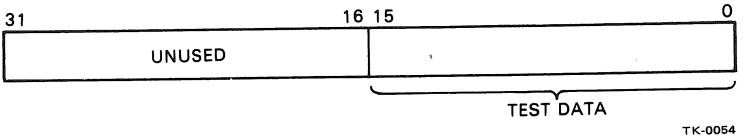
UBA FAILED MAP ENTRY REGISTER, BIT CONFIGURATION



UBA FAILED UNIBUS ADDRESS REGISTER, BIT CONFIGURATION

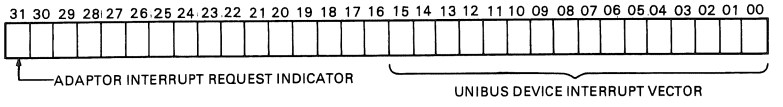


UBA BUFFER SELECTION VERIFICATION REGISTER, BIT CONFIGURATION



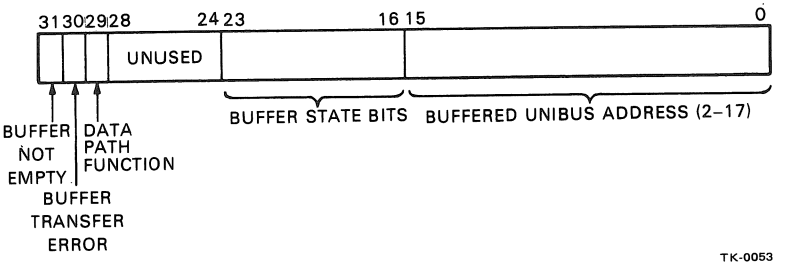
UBA REGISTERS

UBA BR RECEIVE VECTOR REGISTER, BIT CONFIGURATION



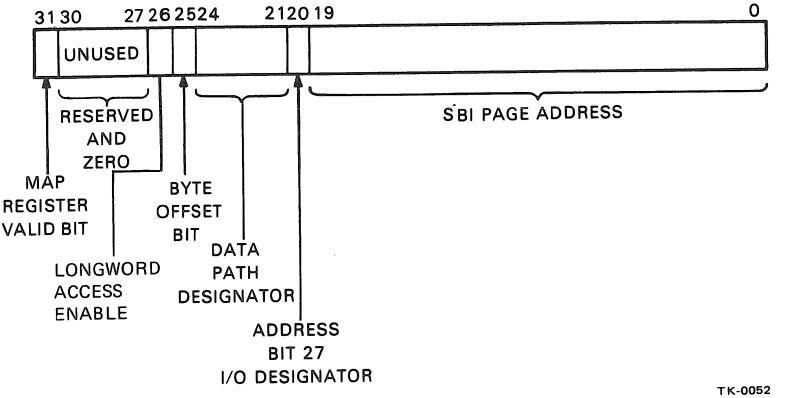
TK-0092

UBA DATA PATH REGISTER, BIT CONFIGURATION



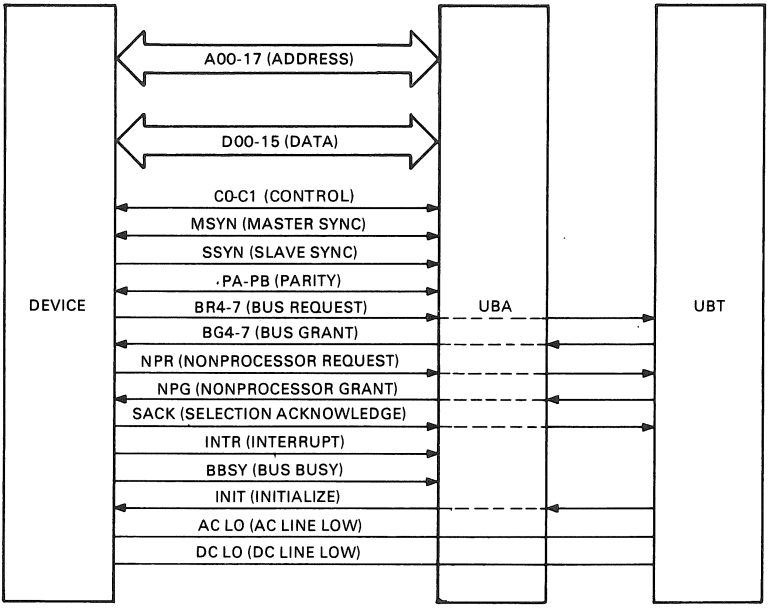
TK-0053

UBA MAP REGISTER, BIT CONFIGURATION



TK-0052

UNIBUS CONFIGURATION



UNIBUS SIGNAL DESCRIPTION

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.

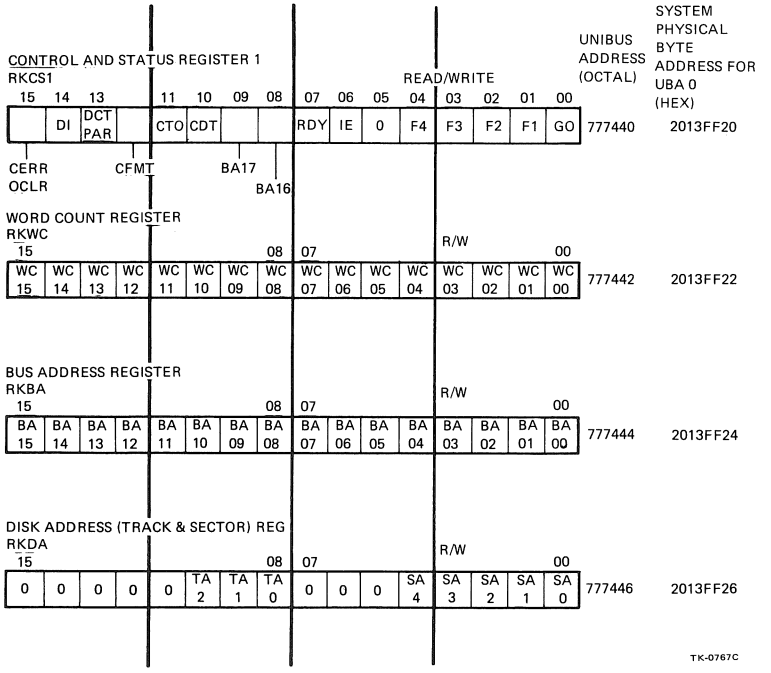
Signal Line	Description
Priority Arbitration Group (Cont)	
Nonprocessor Request (NPR)	This is a bus request from a device for a transfer not requiring CPU intervention (i.e., DMA).
Nonprocessor Grant (NPG)	This is the grant in response to an NPR.
Selection Acknowledge (SACK)	SACK is asserted by a bus-requesting 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.

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

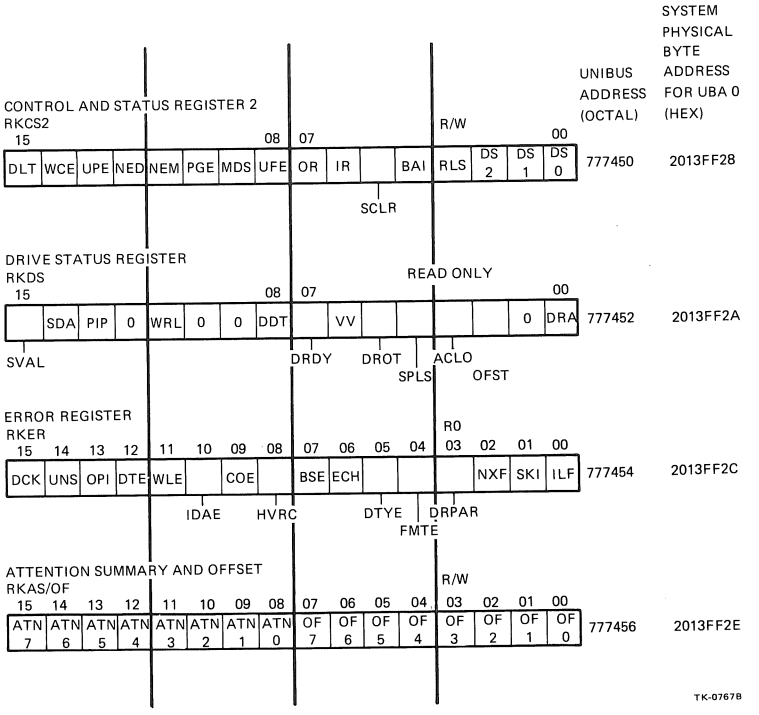
*Pins used by parity control module.

UBA #	DEVICE	UNIBUS ADDRESS (OCTAL)	SBI ADDRESS (HEX)	11/780 PHYSICAL ADDRESS (HEX)	VECTORS (HEX)
0	CR11/CRS1	777160	0804FF9C	2013FE70	98
0	CR11/CRB1	777162	0804FF9C	2013FE72	
0	CR11/CRB2	777164	0804FF9D	2013FE74	
0	LP11/LPST	777514	0804FFD3	2013FF4C	80
0	LP11/LPDR	777516	0804FFD3	2013FF4E	
0	RK611CS1	777440	0804FFC8	2013FF20	(TBS)
0	RK611 WC	777442	0804FFC8	2013FF22	
0	RK611 BA	777444	0804FFC9	2013FF24	
0	RK611 DA	777446	0804FFC9	2013FF26	
0	RK611 CS2	777450	0804FFCA	2013FF28	
0	RK611 DS	777452	0804FFCA	2013FF2A	
0	RK611 ERR	777454	0804FFCB	2013FF2C	
0	RK611 A&O	777456	0804FFCB	2013FF2E	
0	RK611 CYL	777460	0804FFCC	2013FF30	
0	UNUSED	777462	0804FFCC	2013FF32	
0	RK611 DB	777464	0804FFCD	2013FF34	
0	RK611 MR1	777466	0804FFCD	2013FF36	
0	RKECPS	777470	0804FFCE	2013FF38	
0	RKECPT	777472	0804FFCE	2013FF3A	
0	RK611 MR2	777474	0804FFCF	2013FF3C	
0	RK611 MR3	777476	0804FFCF	2013FF3E	
0	DZ11	Floating addresses and vectors			
0	DMC11	according to PDP-11			
0	DR11-B	convention			

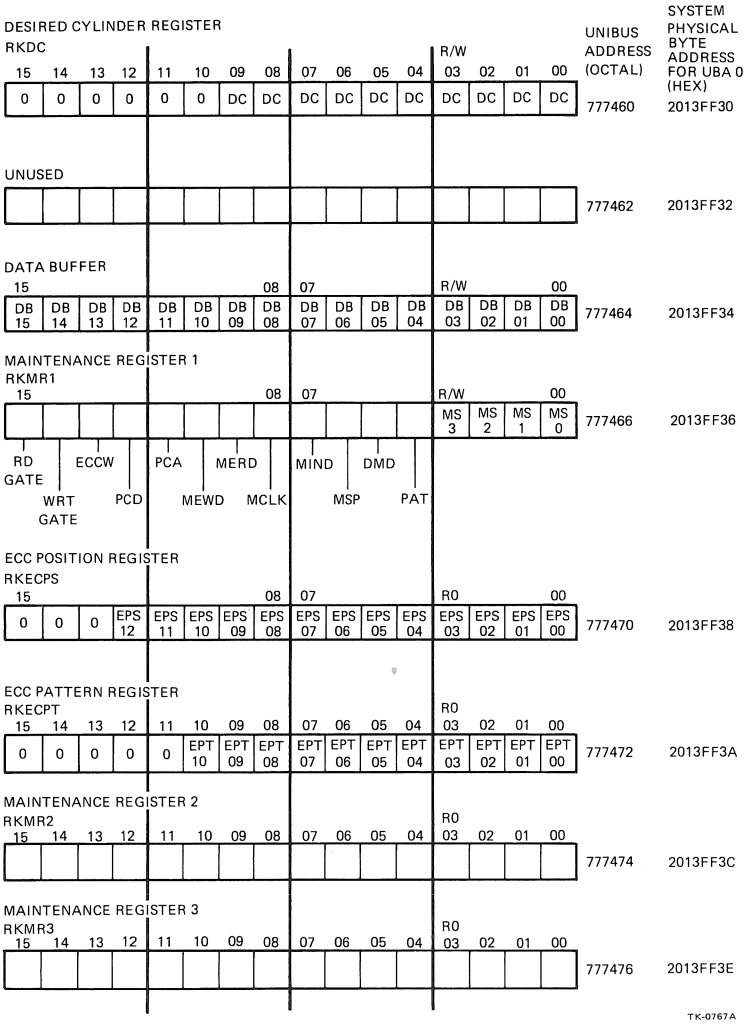
RK611 REGISTER CONTENTS



RK611 REGISTER CONTENTS



RK611 REGISTER CONTENTS



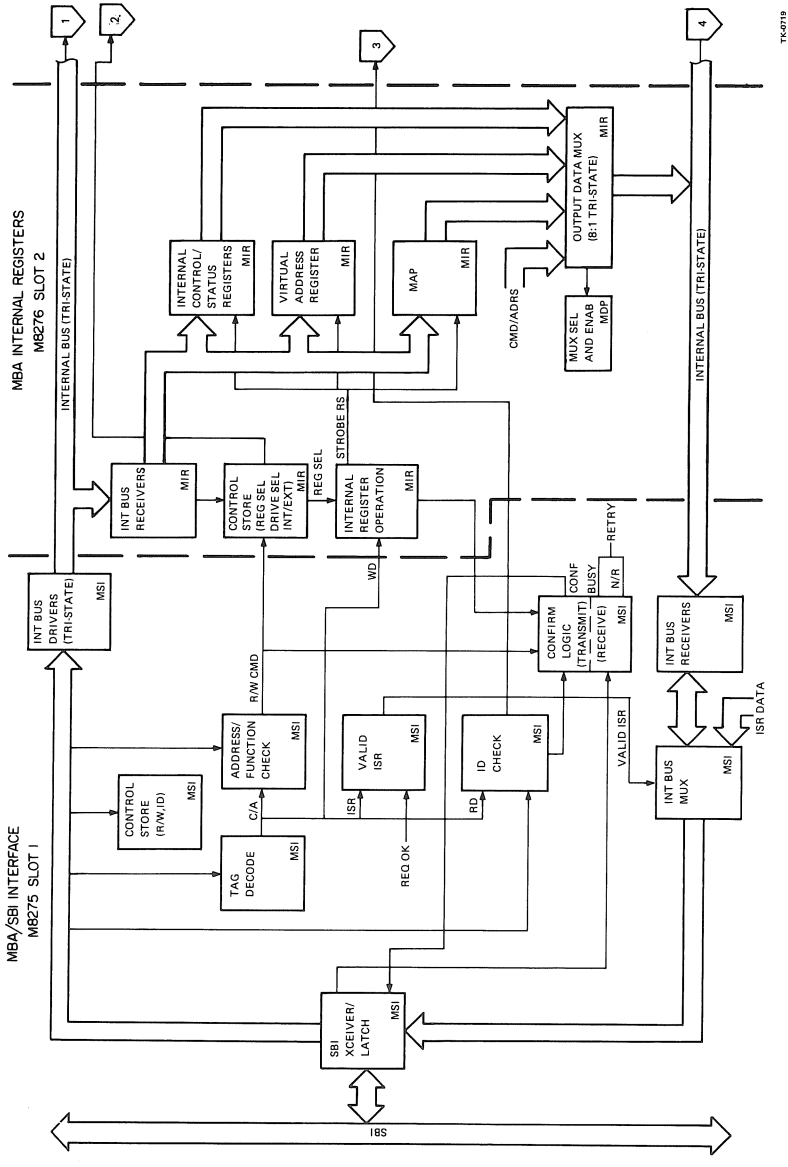
	BYTES: HIGH — MSB 15	14	13	12	11	10	09	08	BYTES: LOW — 07	06	05	04	03	02	01	LSB 00	UNIBUS ADDRESS (OCTAL)	SYSTEM PHYSICAL BYTE ADDRESS FOR UBA 0 (HEX)
CONTROL & STATUS (CSR)	RO TX RDY	RW TX INTR ENAB	RO SILO ALARM	RW SILO ALARM ENAB	NOT USED	RO TX LINE C	RO TX LINE B	RO TX LINE A	RO RX DONE	RW RX INTR ENAB	RW MAST SCAN ENAB	RW CLEAR	RW MAINT	NOT USED	NOT USED	NOT USED	760100**	2013E040**
RECEIVER BUFFER (RBUF)	RO DATA VALID	RO OVRN	RO FRAM ERR	RO PAR ERR	NOT USED	RO RX LINE C	RO RX LINE B	RO RX LINE A	RO RBUF D7	RO RBUF D6	RO RBUF D5	RO RBUF D4	RO RBUF D3	RO RBUF D2	RO RBUF D1	RO RBUF D0	760102**	2013E042**
LINE PARAMETER (LPR)	NOT USED	NOT USED	NOT USED	WO RX ON	WO FREQ D	WO FREQ C	WO FREQ B	WO FREQ A	WO ODD PAR	WO PAR ENAB	WO STOP CODE	WO CHAR LGTH B	WO CHAR LGTH A	WO LINE C	WO LINE B	WO LINE A	760102**	2013E042**
TRANSMIT* CONTROL (TCR)	RO DATA TERM RDY 7	RO DATA TERM RDY 6	RO DATA TERM RDY 5	RO DATA TERM RDY 4	RO DATA TERM RDY 3	RO DATA TERM RDY 2	RO DATA TERM RDY 1	RO DATA TERM RDY 0	RO LINE ENAB 7	RO LINE ENAB 6	RO LINE ENAB 5	RO LINE ENAB 4	RO LINE ENAB 3	RO LINE ENAB 2	RO LINE ENAB 1	RO LINE ENAB 0	760104**	2013E044**
MODEM* STATUS (MSR)	RO CO 7	RO CO 6	RO CO 5	RO CO 4	RO CO 3	RO CO 2	RO CO 1	RO CO 0	RO RING IND 7	RO RING IND 6	RO RING IND 5	RO RING IND 4	RO RING IND 3	RO RING IND 2	RO RING IND 1	RO RING IND 0	760106**	2013E046**
TRANSMIT DATA (TDR)	WO BRK 7	WO BRK 6	WO BRK 5	WO BRK 4	WO BRK 3	WO BRK 2	WO BRK 1	WO BRK 0	WO TBUF 7	WO TBUF 6	WO TBUF 5	WO TBUF 4	WO TBUF 3	WO TBUF 2	WO TBUF 1	WO TBUF 0	760106**	2013E046**

*THE HIGH BYTE OF THE TCR (DATA TERMINAL READY) AND THE MSR ARE NOT USED WITH THE 20MA OPTIONS.

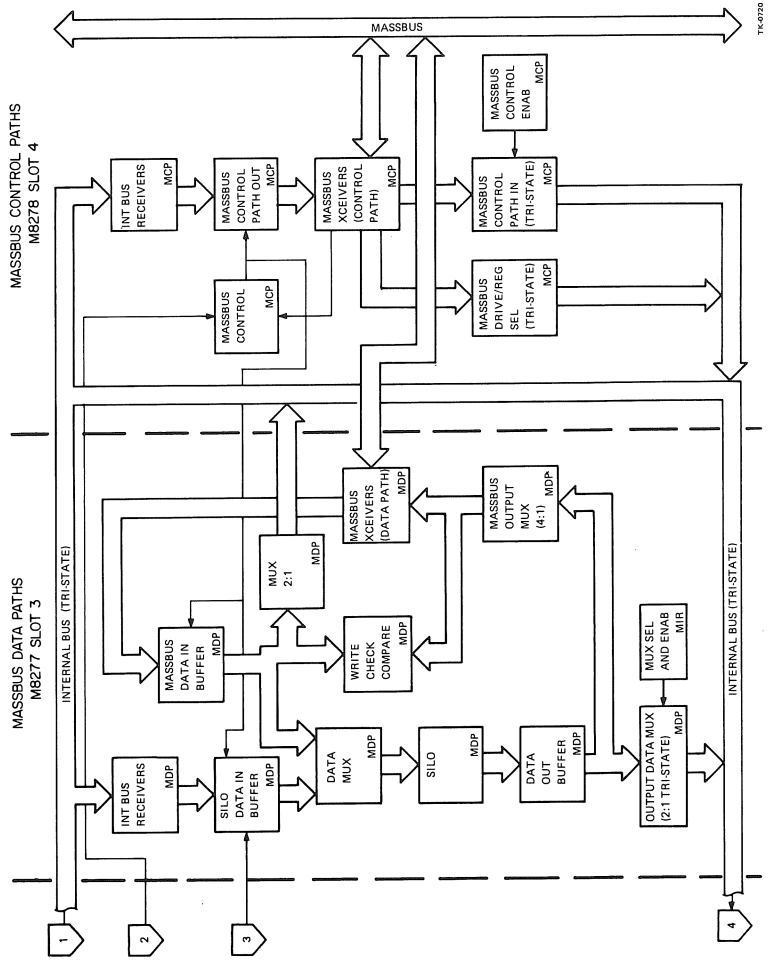
** THESE ADDRESSES REFER TO THE FIRST DZ11 ON UBA 0 ONLY

TK-0765

MBA (RH780) BLOCK DIAGRAM, PART 1



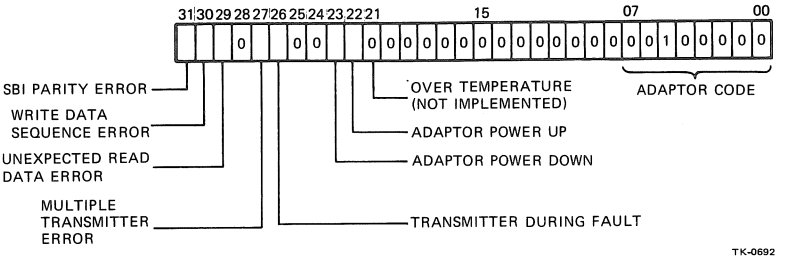
MBA (RH780) BLOCK DIAGRAM, PART 2



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

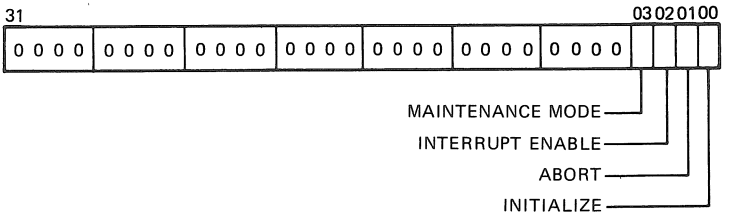
MBA REGISTERS

MBA CONFIGURATION/STATUS REGISTER, BIT CONFIGURATION



TK-0692

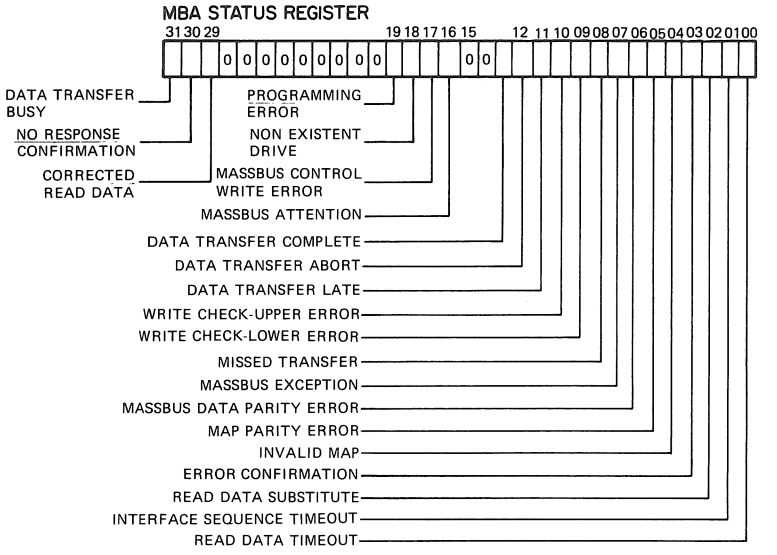
MBA CONTROL REGISTER, BIT CONFIGURATION



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

TK-0693

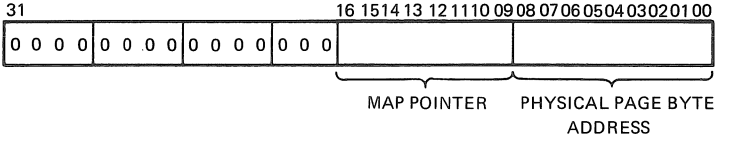
MBA STATUS REGISTER, BIT CONFIGURATION



NOTE: WRITE 1 TO CLEAR BITS IN THIS REGISTER EXCEPT BIT 31 WHICH IS READ ONLY.

TK-0698

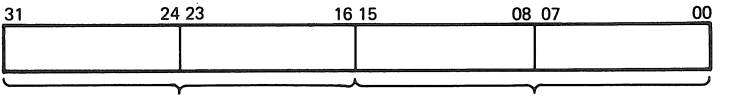
MBA VIRTUAL ADDRESS REGISTER, BIT CONFIGURATION



TK-0696

MBA REGISTERS

MBA BYTE COUNTER REGISTER, BIT CONFIGURATION



MASSBUS BYTE COUNTER (READ ONLY)

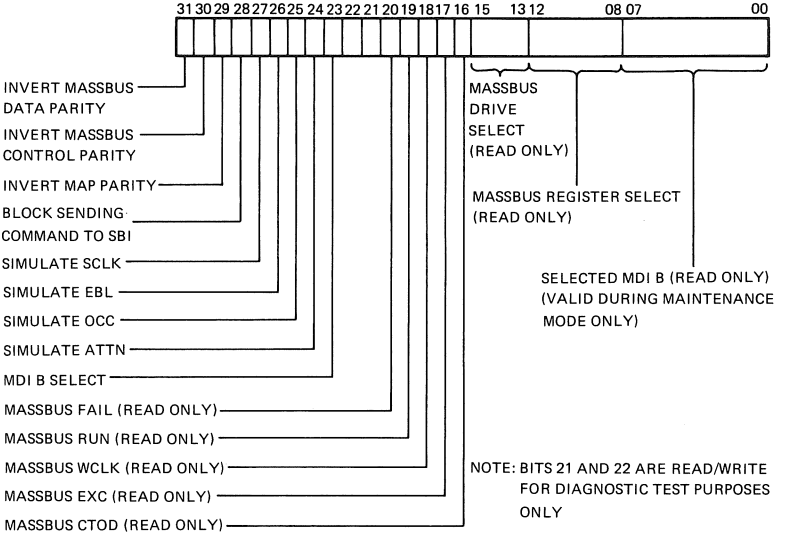
SBI BYTE COUNTER (READ/WRITE)

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

2's COMPEMENT OF THE NUMBER OF BYTES TO BE TRANSFERED

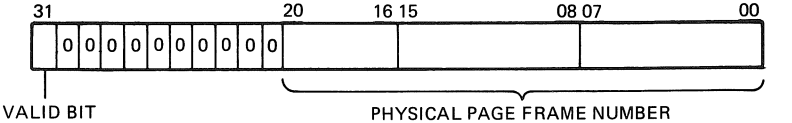
TK-0697

MBA DIAGNOSTIC REGISTER, BIT CONFIGURATION



TK-0694

MBA MAP REGISTER, BIT CONFIGURATION



TK-0715

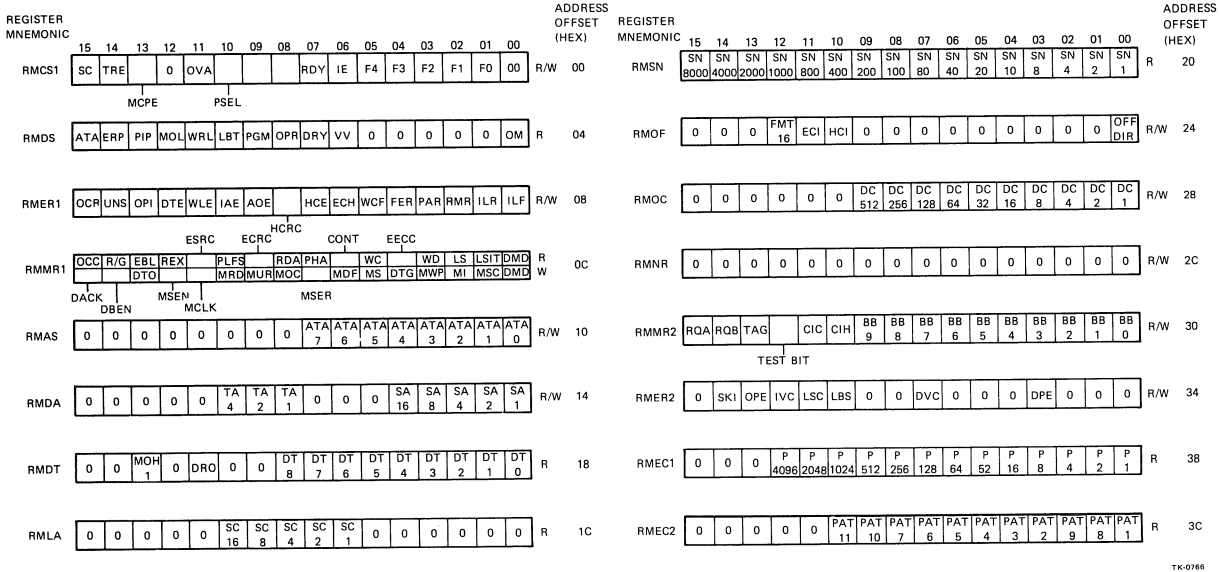
1st MBA Base Address 20010400
2nd MBA Base Address 20012400

REGISTER NUMBER		DRIVE TYPE			DRIVE NUMBER							
HEX	OCTAL	RP (DISK)	RM (DISK)	TE (TAPE)	0	1	2	3	4	5	6	7
0	0	CS1	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

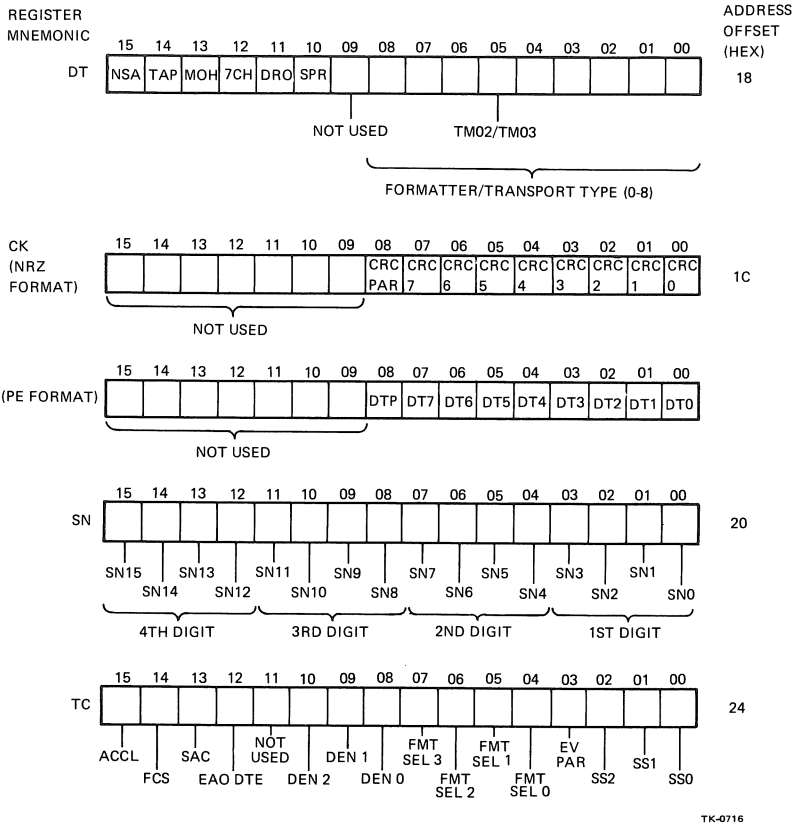
RP05/RP06 REGISTER CONTENTS

REGISTER MNEMONIC	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	ADDRESS OFFSET (HEX)
RPCS1	DVA										F4	F3	F2	F1	F0	GO	00
RPDA	0	0	0	TA 16	TA 8	TA 4	TA 2	TA 1	0	0	0	SA 16	SA 8	SA 4	SA 2	SA 1	14
RPDS	ATA	ERR	PIP	MOL	WRL	LST	PGM	DPR	DRY	VV	0	0	0	0	0	0	04
RPER1	DCK	UNS	OPI	DTE	WLE	IAE	AOE	HCRC	HCE	ECH	WCF	FER	PAR	RMR	ILR	ILF	08
RPAS	0	0	0	0	0	0	0	0	ATA 7	ATA 6	ATA 5	ATA 4	ATA 3	ATA 2	ATA 1	ATA 0	10
RPLA	0	0	0	0	0	SC 4	SC 3	SC 2	SC 1	SC 0	EXT 1	EXT 0	0	0	0	0	1C
RPMR	0	0	0	0	0	0	SBD	0	DEN	ECCE	MWR	MRD	MSCLK	MIND	MCLK	DMD	0C
RPDT	NBA	TAP	MOH	0	DRQ	0	0	DT 8	DT 7	DT 6	DT 5	DT 4	DT 3	DT 2	DT 1	DT 0	18
RPSN	SN 38	SN 34	SN 32	SN 31	SN 28	SN 24	SN 22	SN 21	SN 18	SN 14	SN 12	SN 11	SN 8	SN 4	SN 2	SN 1	20
RPOF	SGCH	0	0	FMT 22	ECCI	HCI	0	0	OFS 7	OFS 6	OFS 5	OFS 4	OFS 3	OFS 2	OFS 1	OFS 0	24
RPDC	0	0	0	0	0	0	DC 10	DC 9	DC 8	DC 7	DC 6	DC 5	DC 4	DC 3	DC 2	DC 1	28
RPCC	0	0	0	0	0	0	CC 10	CC 9	CC 8	CC 7	CC 6	CC 5	CC 4	CC 3	CC 2	CC 1	2C
RPER2	0	0	PLU	0	IXE	NHS	MHS	WRU	ABS	TUF	TDF	RAW	CSU	WSU	CSF	WCU	30
RPER3	OCYL	SKI	OPE	0	0	0	0	0	0	ACL	DCL	35VF	0	0	WAO	DCU	34
RPER1	0	0	0	BLC 4096	BLC 2048	BLC 1024	BLC 512	BLC 256	BLC 128	BLC 64	BLC 32	BLC 16	BLC 8	BLC 4	BLC 2	BLC 1	38
RPEC2	0	0	0	0	0	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	3C

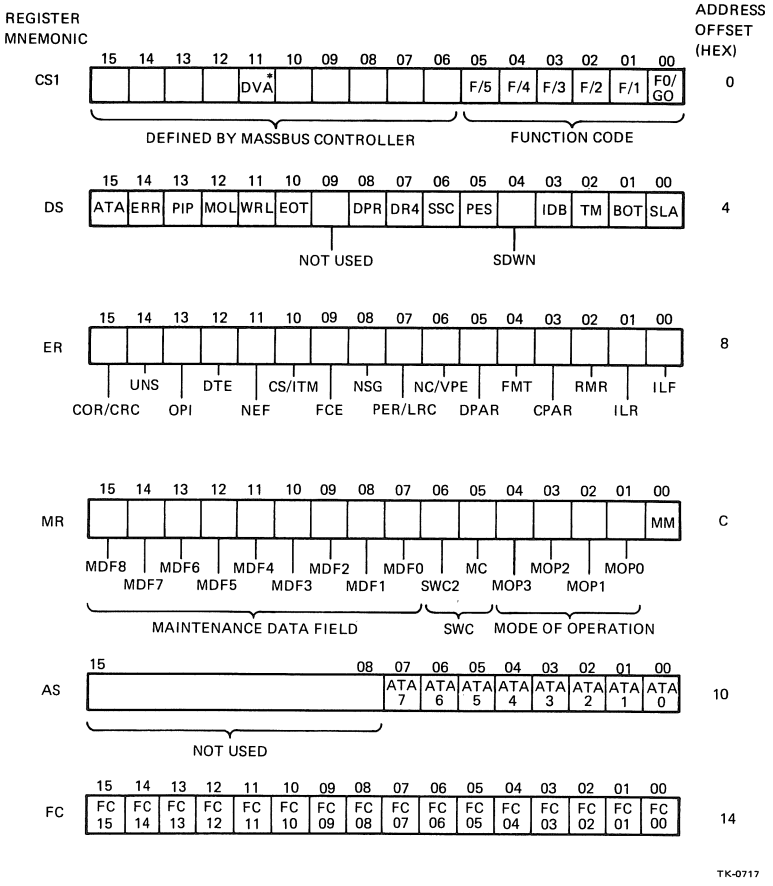
TK-0722


REGISTER MNEMONIC
ADDRESS OFFSET (HEX)
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
RMCS1 SC TRE 0 OVA RDY IE F4 F3 F2 F1 F0 00 R/W 00
MCPE
PSEL
RMDS ATA ERP PIP MOL WRL LBT PGM OPR DRY VV 0 0 0 0 0 OM R 04
RMER1 OCR UNS OPI DTE WLE IAE AOE HCE ECH WCF FER PAR RMR ILR ILF R/W 08
HCRC
ESRC ECRC CONT EECC
RMMR1 OCC R/G EBL REX PLFS RDA PHA WC WD LS LSIT DMD R
DTO MRD MUR MOC MDF MS DTG MWP MI MSC DMD W 0C
DACK DBEN MSEN MCLK MSER
RMAS 0 0 0 0 0 0 0 0 ATA 7 ATA 6 ATA 5 ATA 4 ATA 3 ATA 2 ATA 1 ATA 0 R/W 10
RMDA 0 0 0 0 0 TA 4 TA 2 TA 1 0 0 0 SA 16 SA 8 SA 4 SA 2 SA 1 R/W 14
RMDT 0 0 MOH 1 0 DRO 0 0 DT 8 DT 7 DT 6 DT 5 DT 4 DT 3 DT 2 DT 1 DT 0 R 18
RMLA 0 0 0 0 0 SC 16 SC 8 SC 4 SC 2 SC 1 0 0 0 0 0 0 R 1C
REGISTER MNEMONIC
ADDRESS OFFSET (HEX)
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
RMSN SN 8000 SN 4000 SN 2000 SN 1000 SN 800 SN 400 SN 200 SN 100 SN 80 SN 40 SN 20 SN 10 SN 8 SN 4 SN 2 SN 1 R 20
RMOF 0 0 0 FMT 16 ECI HCI 0 0 0 0 0 0 0 0 0 OFF DIR R/W 24
RMOC 0 0 0 0 0 0 DC 512 DC 256 DC 128 DC 64 DC 32 DC 16 DC 8 DC 4 DC 2 DC 1 R/W 28
RMNR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 R/W 2C
RMMR2 RQA RQB TAG CIC CIH BB 9 BB 8 BB 7 BB 6 BB 5 BB 4 BB 3 BB 2 BB 1 BB 0 R/W 30
TEST BIT
RMER2 0 SKI OPE IVC LSC LBS 0 0 DVC 0 0 0 DPE 0 0 0 R/W 34
RMEC1 0 0 0 P 4096 P 2048 P 1024 P 512 P 256 P 128 P 64 P 52 P 16 P 8 P 4 P 2 P 1 R 38
RMEC2 0 0 0 0 0 PAT 11 PAT 10 PAT 7 PAT 6 PAT 5 PAT 4 PAT 3 PAT 2 PAT 9 PAT 8 PAT 1 R 3C
TK-0766

TM03 REGISTER CONTENTS



TM03 REGISTER CONTENTS



TK-0717

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

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

SECTION 4
CONFIGURATION JUMPERS

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	WCS	
19			
18	M8233 OR M8234	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

PART NO. 3615084

MODULE UTILIZATION MS780		
20	M8214 MSB	
19	M8213 MCN	
18	M8212 MDT	
17	M8211 MAY 0-64 KBYTE	
16	M8211 MAY 64-128 KBYTE	
15	M8211 MAY 128-192 KBYTE	*
14	M8211 MAY 192-256 KBYTE	*
13	M8211 MAY 256-320 KBYTE	*
12	M8211 MAY 320-384 KBYTE	*
11	M8211 MAY 384-448 KBYTE	*
10	M8211 MAY 448-512 KBYTE	*
9	M8211 MAY 512-576 KBYTE	*
8	M8211 MAY 576-640 KBYTE	*
7	M8211 MAY 640-704 KBYTE	*
6	M8211 MAY 704-768 KBYTE	*
5	M8211 MAY 768-832 KBYTE	*
4	M8211 MAY 832-896 KBYTE	*
3	M8211 MAY 896-960 KBYTE	*
2	M8211 MAY 960-1024 KBYTE	*
1	M9040 TRM	*

* OPTIONAL IF NOT INSTALLED USE BLANK MODULE 7014103

PART NO. 3614746

DW780 MODULE UTILIZATION					
6	5	4	3	2	1
BLANK MODULE	BLANK MODULE	UAI M8273	UMD M8272	UCB M8271	USI M8270

PART NO. 3614748

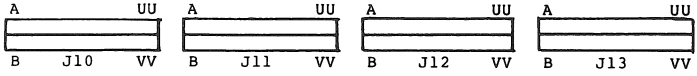
RH780 MODULE UTILIZATION					
6	5	4	3	2	1
BLANK MODULE	BLANK MODULE	MCP M8278	MDP M8277	MIR M8276	MSI M8275

PART NO. 3614747

11780 REV		
OPTION	TR#	REV
KA780		
MS780		
DW780		
RH780		

PART NO. 36-14849-00

KA780 TR, SYS.ID REGISTER JUMPERS



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 F0M2 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	-----

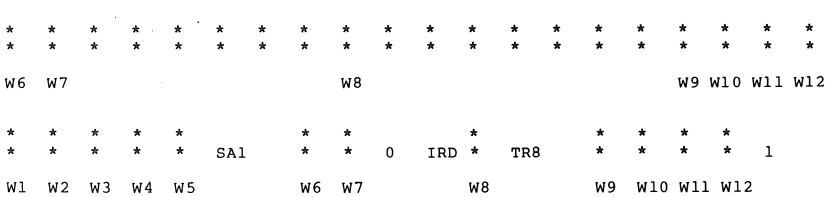
System ID Register
Remove Jumper to Assert Bit

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

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

PCS
SLOT 22
J12

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



TR Arbitration Level

SIGNAL NAME	TR SEL 8	TR SEL 4	TR SEL 2	TR SEL 1
TR#	W9	W10	W11	W12
1	--	--	--	-- [1,2]
2	--	--	--	I
3	--	--	I	--
4	--	--	I	I
5	--	I	--	--
6	--	I	--	I
7	--	I	I	--
8	--	I	I	I
9	I	--	--	--
10	I	--	--	I
11	I	--	I	--
12	I	--	I	I
13	I	I	--	--
14	I	I	--	I
15	I	I	I	--

WIRE WRAP F20 K2 to

F20 C1
F20 D1
F20 E1
F20 F2
F20 H2
F20 J1
F20 J2
F20 M1
F20 N1
F20 P1
F20 P2
F20 S2
F20 T2
F20 U1
F20 U2

Inhibit ROM Decode

W8=I — No response[2] confirmation for Read to ROM Address Space

W8=-- — Read to ROM[1] Address Space receive Normal confirmation

NOTE: When installing an MS780A and an MS780C on the same VAX-11/780 system, the MS780C must be the first memory (i.e., lowest TR).

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

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

Memory Array Boards (M8214, M82823)

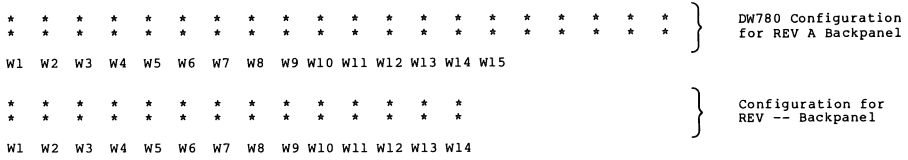
SYN BITS 7-3	BYTE	SYN BITS 2-0 (BIT IN ERROR) 111	110	101	100	011	010	001	000
	LOWER								
00*11	L00	7	6	5	4	3	2	1	0
01*01	L01	15	14	13	12	11	10	9	8
01*10	L02	23	22	21	20	19	18	17	16
01*11	L03	31	30	29	28	27	26	25	24
	UPPER								
11*01	U00	7	6	5	4	3	2	1	0
10*10	U01	15	14	13	12	11	10	9	8
10*11	U02	23	22	21	20	19	18	17	16
11*00	U03	31	30	29	28	27	26	25	24
00000	CHECK				C2		C1	C0	
00001	CHECK								C3
00010	CHECK								C4
00100	CHECK								C5
01000	CHECK								C6
10000	CHECK								C7

* DON'T CARE ON THE VALUE OF THIS BIT (USED ONLY TO KEEP AN ODD NUMBER OF CHECK BITS PRESENT AT ALL TIMES).

ERROR LOGOUT

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

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



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 D0142 to
TR#	W3	W4	W5	W6	
1	--	--	--	--	F01C1
2	I	--	--	--	F01D1
3	--	I	--	-- '	F01E1
4	I	I	--	--	F01F2
5	--	--	I	--	F01H2
6	I	--	I	--	F01J1
7	--	I	I	--	F01J2
8	I	I	I	--	F01M1
9	--	--	--	I	F01N1
10	I	--	--	I	F01P1
11	--	I	--	I	F01P2
12	I	I	--	I	F01S2
13	--	--	I	I	F01T2
14	I	--	I	I	F01U1
15	--	I	I	I	F01U2

Unibus Address Space Select

SIGNAL NAME	USID ADAPTOR 0 L	USID ADAPTOR 1 L
ADAPTOR#	W1	W2
0	--	-- '
1	I	--
2	--	I
3	I	I

INTERRUPT LEVEL SELECTION SIGNAL NAME

	UAIF SBI PRI JMP 0 L	UAIF SBI PRI JMP 1 L
ISR#	W7	W8
4	--	-- '
5	I	--
6	--	I
7	I	I

' Normal for 1st DW780

RH780 (MBA) BACKPANEL JUMPER CONFIGURATION

TR Arbitration Level

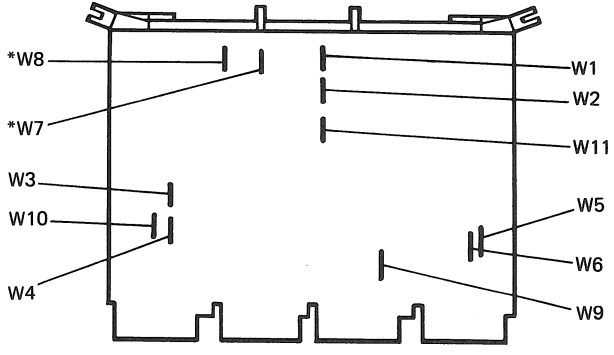
SIGNAL NAME	MBA TR SEL D	MBA TR SEL C	MBA TR SEL B	MBA TR SEL A
TR#	W1	W2	W3	W4
1	--	--	--	--
2	--	--	--	I
3	--	--	I	--
4	--	--	I	I
5	--	I	--	--
6	--	I	--	I
7	--	I	I	--
8	--	I	I	I '
9	I	--	--	--
10	I	--	--	I
11	I	--	I	--
12	I	--	I	I
13	I	I	--	--
14	I	I	--	I
15	I	I	I	--

Interrupt Level Selection

WIRE WRAP BUS SBI TRXX L from F02F1 to	SIGNAL NAME	MBA INTR CODE 1 H	MBA INTR CODE 0 H
BR#			
F02C1	4	--	--
F02D1	5	--	I '
F02E1	6	I	--
F02F2	7	I	I
F02H2			
F02J1			
F02J2			
F02M1			
F02N1			
F02P1			
F02P2			
F02S2	W7 - W12 SPARES		
F02T2			
F02U1			
F02U2			

' Normal for 1st RH780

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



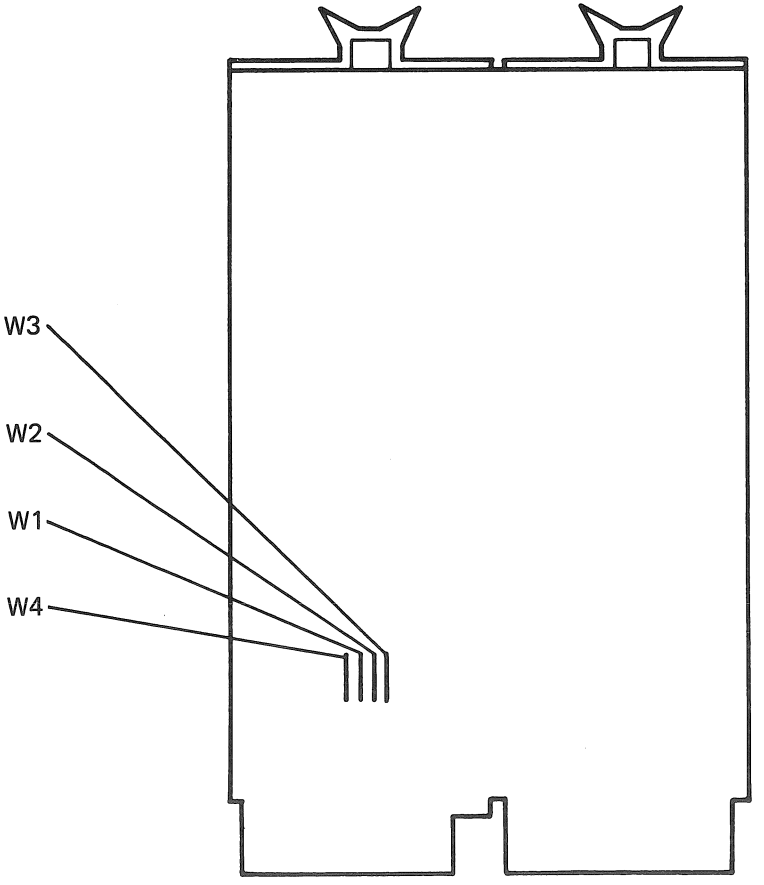
M7264 ETCH REV. E (AND LATER)

*FACTORY CONFIGURED DO NOT CHANGE (W7 & W8)

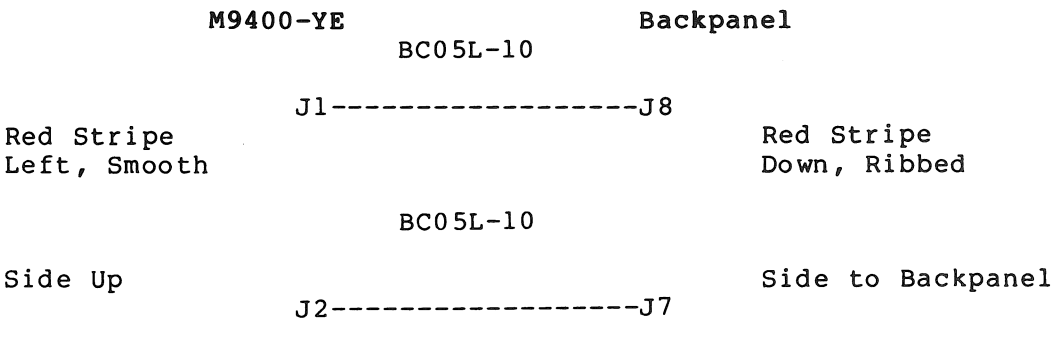
TK-0700

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:

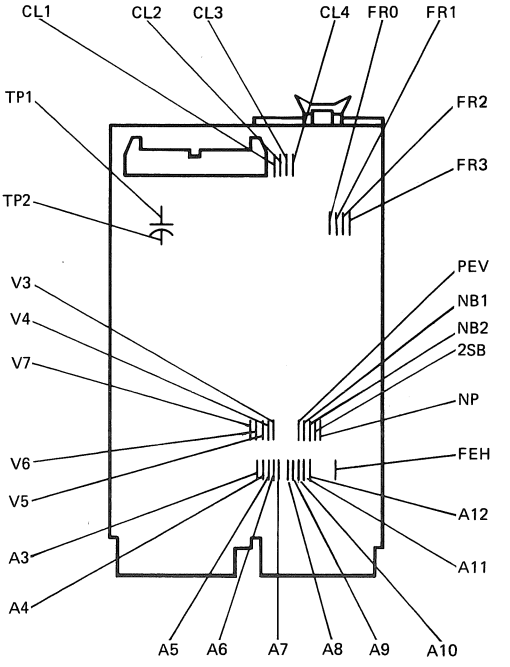
Address: 177170-177172
Vector: 264
Ribbon cable should be installed with
red stripe toward center of module.

DLV11 JUMPER CONFIGURATION

JUMPER	IN/OUT	RESULT
	–	NO PARITY
2SB	I	1 STOP BIT
NB2	–	8 DATA BITS
NB1	–	8 DATA BITS
PEV	X	DON'T CARE PARITY EVEN/ODD
FEH	X	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=I, V6=I, V5=–, V3=I

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



TK-0701

DLV11-E JUMPER CONFIGURATION

DLV IIE

R3 R2 R1 R0
I – I –

T3 T2 T1 T0
– – – I

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

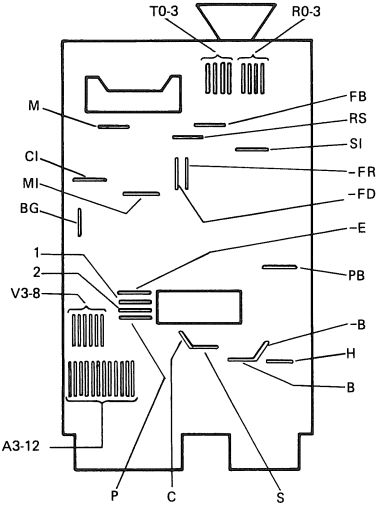
V8 V7 V6 V5 V4 V3
– I I – – I

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
– – –

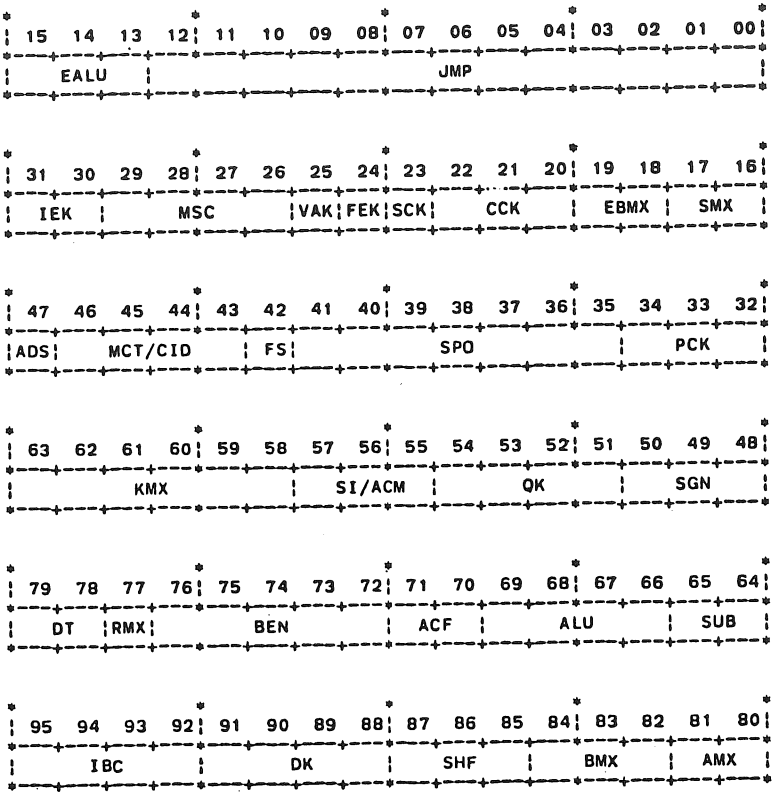
TK-0726



TK-0718

SECTION 5
CONTROL STORE

CONTROL STORE FIELD MAP



MICROCODE ROUTINES WHICH SUPPORT CONSOLE SOFTWARE, STARTING ADDRESSES

```
        "CONSOLE MICRO-CODE"


        ;MICRO-CODE ROUTINES TO SUPPORT CONSOLE SOFTWARE.
        ;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	

MICROCODE BRANCH ENABLE FUNCTIONS

MICRO 31(253) STAR Microcode: PCS 01, FPLA 04, WCS113. FEB 16, 78
Branch Enable Functions

"Branch Enable Functions" 31-January-77

BEN	Name	UPC<02>	UPC<01>	UPC<00>
0	NOP	0	0	0
1	Z	0	0	ALU Z
2	ROR	LA<01>	PSL<C>	LA<00>
3	C31	0	ALU C31	0
4				
5				
6	ACCelerator	ACC UB2	ACC UB1	ACC UB0
7				
8 VAX	DATA TYPE 0= Normal 2= Field Src 1= Q + D 3= Addr Src	Read + Modify	ASRC + VSRC	ASRC + Q + D
8 -11	END DP1	Read + Modify	0 Class	J Class + DM27
9 VAX	IR2-1	0	IR<2>	IR<1>
9 -11	PC Modes	0	SM or DM 47 + 57	Dst R .eq. PC
A	REI	Mode .lt. ASTLVL		
B	IB TEST 0= TB Miss 2= Stall 1= Error 3= Data OK	0	IB running	IB ERROR + DATA VALID
C	MUL	SC.ne.0	D<01>	D<00>
D	SIGNS	Q<31>	D.ne.0	D<31>
E	INTERRUPT	AC Low	Internal Interrupt	Interrupt Request
F	Decimal	0	D<7:0> 30-39	D<3:0>= 08 + 0D

MICROCODE BRANCH ENABLE FUNCTIONS

BEN	Name	UPC<03>	UPC<02>	UPC<01>	UPC<00>
10	uTrap Vector	uVECT<3>	uVECT<2>	uVECT<1>	uVECT<0>
11	Last Reference	-PSL<FPD>	Nested Error	Wr Chk* -Intlk	-Read+ Intlk
12	EALU CC	EALU N	EALU Z	SC.ne.0	Sign Src
13					
14	SC 0= Zero 2= 1-31 1= Neg 3= .gt.31	0	SC<9:8> .ne.0	SC.gt.0	SC<9:5> .ne.0
15	ALU1-0 (previous cycle)	Rlog Empty	ALU<1:0> .eq.0	ALU<01>	ALU<00>
16	STATE7-4	STATE<7>	STATE<6>	STATE<5>	STATE<4>
17	STATE3-0	STATE<3>	STATE<2>	STATE<1>	STATE<0>
18	D Bytes	D<31:24> .ne.0	D<23:16> .ne.0	D<15:8> .ne.0	D<7:0> .ne.0
19	D3-0	D<03>	D<02>	D<01>	D<00>
1A	PSL CC	PSL<N>	PSL<Z>	PSL<V>	PSL<C>
1B	ALU CC	ALU N	ALU Z	IR<0>	ALU C31

BEN	Name	UPC<04>	UPC<03>	UPC<02>	UPC<01>	UPC<00>
1C	PSL Mode	-VAMX<31>	-VAMX<30>	-Console Mode	-PSL<IS>* -PSL<CM>	Kernel Mode
1D	Translation Test	PTE - Valid	Data Aligned	0	TB Miss + Access Viol	TB Miss + 1st Modify
1E						
1F						

MICROTRAP VECTORS

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

HOW TO READ THE MICROCODE

(1) Field Definitions

These occur at the beginning of the listing, in the source file DEF.MIC. They have the form:

```
SYMBOL/=J,K,L,M
```

The first parameter (J) is meaningful only when "D" is specified as the default mechanism, and in that case, gives the default value of the field in hexadecimal.

The second parameter (K) gives the field size in (decimal) number of bits.

The third parameter (L) gives the field position in decimal as the bit number of the rightmost bit of the field. Bits are numbered from 0 on the right.

The fourth parameter (M) is optional, and selects a default mechanism for the field. The legal values of this parameter are the characters "D", or "+".

"D" means J is the default value of the field if no explicit value is specified.

"+" is used on the jump address field to specify that the default jump address is the address of the next instruction assembled (not, in general, the current location +1).

In general, a field corresponds to the set of bits which provide select inputs for mixers or decoders, or controls for ALU's.

Examples:

```
AMX/=0,2,20,D
```

The microcode field which controls AMX is two bits wide and the rightmost bit is shown in the listing as bit 20 of the micro.nstruction. If no value is specifically requested for the field, the microassembler will ensure that the field is 0.

```
ALU/=0,4,25
```

The field which controls alu is 4 bits wide, ending on bit 25. The fourth parameter of the field is omitted, so the field is available to the microassembler (if no value is explicitly called out for the field) for modification.

(2) Value Definitions

Following a field definition, symbols may be created in that field to correspond to values of the field. the form is:

```
SYMBOL=N
```

"N" is, in hex, the value of symbol when used in the field.

examples:

```
ALU/=0,4,25     ;Field definition in which following symbols exist
XOR=7
A+B=8
```

Here the symbols "XOR" and "A+B" are defined for the "ALU" field. To the assembler, therefore, writing "ALU/XOR" means put the value 7 into the 4-bit field ending on bit 25 of the microword. The symbols are chosen for mnemonic significance, of course, so one reading the microcode would interpret "ALU/XOR" as "the output of ALU shall be the exclusive or of its A and B inputs". Similiarly, "ALU/A+B" is read as "ALU produces the sum of A and B".

```
SCK/=0,1,23,D   ;field definition for following symbols
NOP=0
LOAD=1
```

Here the symbols "NOP" and "LOAD" are defined for the field named "SCK", which controls the clocking of the SC register. We could write SCK/NOP in every microinstruction in which we did not want the SC to change, but to simplify things, we use the default mechanism, which ensures that unless a microinstruction explicitly specifies a change to SC (by SCK/LOAD), the assembler will make the value of this field 0.

(3) Label Definitions

A micro instruction may be labelled by a symbol followed by colon preceding the microinstruction definition. The address of the microinstruction becomes the value of the symbol in the field named "J". example:

```
FOO:    J/FOO
```

This is a microinstruction whose "J" field (jump address) contains the value "FOO". It also defines the symbol "FOO" to be the address of itself. Therefore, if executed by the microprocessor, it would loop on itself.

(4) Comments

A semicolon anywhere on a line causes the rest of the line to be ignored by the assembler. This text is an example of comments.

(5) Microinstruction Definition

A word of microcode is defined by specifying a field name, followed by slash (/), followed by a value. The value may be a symbol defined for that field, a hex digit string, or a decimal digit string (distinguished by the fact that it is terminated by a period). Several fields may be specified in one microinstruction by separating field/value specifications with commas. example:

```
AMX/LA,BMX/D,ALU/A-B
```

The field named "AMX" is given the value named "LA" (to cause the mixer on the A side of ALU to select LA), field "BMX" has value "D", field "ALU" has value "A-B".

(6) Continuation

The definition of a microinstruction may be continued onto two or more lines by breaking it after any comma. In other words, if the last non-blank, non-comment character on a line is a comma, the instruction specification is continued on the following line. example:

```
AMX/LA,BMX/D,           ;Select LA & D as ALU inputs
ALU/A-B                 ;Select ALU to perform A-B
```

By convention, a blank line and a line of hyphens appears between microinstructions, to make it easier for the reader to distinguish continuation from separate microinstructions.

(7) Macros

A macro is a symbol whose value is one or more field/value specifications and/or macros. A macro definition is a line containing the macro name followed by a quoted string which is the value of the macro. example:

```
D_LA-D          "AMX/LA,BMX/D,ALU/A-B,D_ALU"
```

The appearance of a macro in a microinstruction definition is equivalent to the appearance of its value. Macros may have parameters, enclosed in square brackets ("[" and "]"). The definition of a macro with parameters includes paired brackets to indicate where the parameters should go, and uses "@" followed by a decimal digit string to indicate which symbols in the macro body should be replaced by the parameters:

```
RC[]_D+K[]      "AMX/D,KMX/@2,BMX/KMX,ALU/A+B,SPO.RC/@1"
```

This macro indicates that the first parameter (selected by @1) should be used as the value in the "SPO.RC" field, and the second parameter as the value of the "KMX" field. A typical use of this macro might look like:

```
RC[T1]_D+K[34]
```

In this case, the expansion would be "AMX/D,KMX/34,BMX/KMX,ALU/A+B,SPO.RC/T1"

Macros for various functions are defined in "MACRO.MIC".

(8) Pseudo Ops

The micro assembler has the following pseudo-operators:

.UCODE and .DCODE select the ram into which subsequent microcode will be loaded, and therefore the field definitions and macros which are meaningful in subsequent microcode.

.TITLE defines a string of text to appear in the page header, and

.TOC defines an entry for the table of contents at the beginning.

.PAGE starts a new page of listing, and creates a .TOC entry if a quoted string follows the pseudo-op.

.SET defines the value of a conditional assembly parameter,

.CHANGE redefines a conditional assembly parameter,

.DEFAULT assigns a value to an undefined parameter.

.IF enables assembly if the value of the parameter is not zero,

.IFNOT enables assembly if the parameter value is zero, and

.ENDIF re-enables assembly if suppressed by the parameter named.

.RTOL enables bits numbered from 0 on the right of the microinstruction.

.HEXADECIMAL enables radix to be 16 instead of default radix 8.

.REGION defines prefered parts of the .UCODE space.

.CREF and .NOCREF enable and disable the collection of cross-reference information on symbol usage.

.LIST and .NOLIST enable and disable output listing.

.BIN and .NOBIN enable and disable leaving room at the left margin for binary output.

.MACHINE selects microassembler features needed for special microproccessors.

(9) Location Control

A microinstruction "labelled" with a number is assigned to that address.

The character "=" at the beginning of a line, followed by a string of 0's, 1's, and/or *'s, specifies a constraint on the address of following microinstructions. The number of characters in the constraint string (excluding the "=") is the number of low-order bits constrained in the address. The microassembler attempts to find an unused location whose address has 0 bits in the positions corresponding to 0's in the constraint string and 1 bits where the constraint has 1's. Asterisks denote "don't care" bit positions.

If there are any 0's in the constraint string, the constraint implies a block of <2**N> microwords, where N is the number of 0's in the string. All locations in the block will have 1's in the address bits corresponding to 1's in the string, and bit positions denoted by *'s will be the same in all locations of the block.

In such a constraint block, the default address progression is counting in the "0" positions of the constraint string, but a new constraint string occuring within a block may force skipping over some locations of the block. Within a block, a new constraint string does not change the pattern of default address progression, it merely advances the location counter over those locations. The microassembler will later fill them in.

A null constraint string ("=" followed by anything but "0", "1", or "*") serves to terminate a constraint block.

examples:

=0

This specifies that the low-order address bit must be zero--The microassembler finds an even-odd pair of locations, and puts the next two microinstructions into them.

=11

This specifies that the two low-order bits of the address must both be ones. Since there are no 0's in this constraint, the assembler finds only one location meeting the constraint.

=0*****

This specifies a pair of addresses, the first having a zero in the "20" bit, and the second having a one in that position, but all other bit positions the same.

```
;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
.CREF                   ;ENABLE CREF OF ALU FUNCTIONS

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
.NOCREF

AMX/=0,2,80             ;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

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=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/=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 UKMX
        WRITE.SC=0D                     ;WRITE REG SELECTED BY SC
        WRITE.KMX=0F                    ;WRITE REG SELECTED BY UKMX

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
        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)
```

```
        .FFFB=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)
        .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/=3E,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.WCHK=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.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/=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.DR.1=4              ;4      SRC R .DR. 1    SRC R .DR. 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

                        ; 2014  .BIN  ;RE-ENABLE LISTING SPACE FOR BINARY OUTPUT
                        ; 2015  .CREF ;RE-ENABLE CROSS REFERENCE
                        ; 2016
```

```
                                                                ;October 11, 1976

                          ---  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  N  Y  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)
```

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

ALU_0(A)          "AMX/RAMX.OXT,DT/LONG,ALU/A"
ALU_0[]D          "ALU/@1,AMX/RAMX.OXT,LONG,BMX/RBMX,RBMX/D"
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+D           "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A+B"
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+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          "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-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           "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+Q+1         "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B+1"
ALU_-1            "AMX/RAMX.OXT,DT/LONG,ALU/NOTA"
ALU_D             "RAMX/D,AMX/RAMX,ALU/A"
ALU_D.OXT[]       "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A"
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[]+K[]   "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_D.OXT[]-K[]   "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B"
ALU_D.OXT[]+LC    "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/LC"
ALU_D.OXT[]+Q     "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/RBMX,RBMX/Q"
ALU_D.OXT[]-Q     "RAMX/D,AMX/RAMX.OXT,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.OXT,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.OXT[]      "RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A"
ALU_Q.OXT[].ANDNOT.K[]   "ALU/ANDNOT,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.OXT[].OR.K[]       "ALU/OR,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.OXT[].OR.D         "ALU/OR,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/RBMX,RBMX/D"
ALU_Q.OXT[]+D    "ALU/A+B,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RBMX/D,RAMX/Q"
ALU_Q.OXT[]+D+1  "ALU/A+B+1,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RAMX/Q,RBMX/D"
ALU_Q.OXT[]+K[] "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.OXT[]-K[] "ALU/A-B,AMX/RAMX.OXT,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.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
D_0+K[]+1          "AMX/RAMX.0XT,DT/LONG,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_0+LC+1           "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,DK/SHF"
D_0-Q              "AMX/RAMX.0XT,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.0XT[]          "RAMX/D,AMX/RAMX.0XT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_D.0XT[].ANDNOT.K[]      "RAMX/D,AMX/RAMX.0XT,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,SPD.R/LOAD.LC,SPD.RC/@2,BMY/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,SPD.R/LOAD.LC,SPD.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,SPD.R/LOAD.LC,SPD.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,SPD.R/LOAD.LC,SPD.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,AMY/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.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-K[]             "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-LC              "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_0+MASK+1          "AMX/RAMX.OXT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,QK/SHF"
Q_0+PC.RLOG         "AMX/RAMX.OXT,DT/LONG,BMX/PC,ALU/A+B.RLOG,SHF/ALU,QK/SHF"
Q_0-Q               "AMX/RAMX.OXT,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.OXT[]           "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/ALU,QK/SHF"
Q_D.OXT[]+K[].LEFT  "RAMX/D,AMX/RAMX.OXT,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.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[].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.OR.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.OXT,DT/LONG,ALU/A,SHF/ALU"
R[]_0-D            "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-K[]          "AMX/RAMX.OXT,DT/LONG,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-LB           "AMX/RAMX.OXT,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_0+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/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"


;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.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.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(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.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_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.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.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]"
;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"

.TOC    "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,CLK.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"

.TOC    "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"    ;,J5/1B"
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"   ;,J4/07"
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"
```

```
;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
        K3FF=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
        MG=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=0E               ;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
        R0=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 3(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
        MCI.MP1=0A              ;LOAD M'CAND INTEGER&MULTIPLIER HIGH FRACTION
        MCI=0B                  ;LOAD MULTIPLICAND INTEGER
        MP=0C                   ;LOAD MULTIPLIER
        MP0=0D                  ;LOAD LOWER HALF OF MULTIPLIER
        MP1=0E                  ;LOAD MULTIPLIER, HIGH FRACTION
        INIT=0F                 ;CONTROL INITIALIZATION

;       ****************************************
;       *                                      *
;       *              BEN TABLE               *
;       *                                      *
;       ****************************************

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

```
;1      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,EAC/PR.LA"
LA&PR_PR-K              "AMXC/PR,EALUC/A-B,BMXC/#80,BSC/NH,EAC/PR.LA"
LA&SA_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"
MCONT            "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_FAD           "FADC/A.B,BSC/FAL.X,NRC/LD"
NR_PROD          "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/NK,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/K3FF"
PR_XR+LB	"EAC/LDPR,AMXC/LB,BMXC/XR,EALUC/A+B"
RR_NR	"LRR/LD.RR"
RSV	"MSC/MO"
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"

SECTION 6
TROUBLESHOOTING TOOLS/DIAGNOSTICS

CONSOLE HELP FILE

```
VAX-11/780 CONSOLE HELP FILE    REV. 4  8-NOV-1977

GENERAL:        <ADDRESS> IS A <NUMBER>, OR ONE OF THE FOLLOWING
                SYMBOLIC <ADDRESSES>(ONLY FOR EXAMINE AND DEPOSIT COMMANDS)
                '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> IS A STRING OF DIGITS IN THE CURRENT DEFAULT RADIX,
                OR A STRING OF DIGITS PREFIXED WITH A DEFAULT RADIX OVERRIDE(%O
                FOR OCTAL, %X FOR HEX)

        ALL COMMANDS ARE TERMINATED BY CARRIAGE RETURN
 'EXAMINE <ADDRESS>'            -DISPLAYS CONTENTS OF <ADDRESS>
 'DEPOSIT <ADDRESS> <DATA>'     -DEPOSITS <DATA> TO <ADDRESS>
                USE A QUALIFIER AFTER THE COMMAND NAME TO SPECIFY
                THE PROPER ADDRESS SPACE TO USE:
                '/P'    FOR PHYSICAL MEMORY(THE DEFAULT)
                '/V'    FOR VIRTUAL MEMORY
                '/I'    FOR INTERNAL(PROCESSOR) REGISTERS
                '/G'    FOR GENERAL REGISTERS 0 THRU F(R0 THRU PC)
                '/VB'   FOR VBUS REGISTERS
                '/IO'   FOR IOBUS REGISTERS
        EXAMPLE: TO EXAMINE VIRTUAL ADDRESS 10245, THE SHORTEST
                UNIQUE COMMAND STRING IS: 'E/V 10245'

 'EXAMINE IR'                   -EXAMINES INSTRUCTION REGISTER(IR). DISPLAYS
                                 OP-CODE, SPECIFIER, & EXECUTION POINT COUNTER
 'START <ADDRESS>'              -INITIALIZES THE CPU,DEPOSITS <ADDRESS>
                                 TO THE PC, ISSUES A CONTINUE TO THE ISP.
 'CONTINUE'                     -ISSUES A CONTINUE TO THE ISP.
 'HALT'                 -HALTS THE ISP
 'BOOT'                 -BOOTS THE CPU FROM DEFAULT DEVICE
 'INITIALIZE'                   -INITIALIZES THE CPU
```

CONSOLE HELP FILE

```
'SHOW'                    -DISPLAYS CONSOLE AND CPU STATE
'SHOW VERSION'            -DISPLAYS VERSIONS OF MICROCODE AND CONSOLE
'TEST'                    -RUNS MICRO-DIAGNOSTICS
'TEST/COM'                        -CALLS MICRO-DIAGNOSTIC MONITOR,AWAITS COMMANDS
'UNJAM'                           -UNJAMS THE SBI
'SET STEP BUS'            -ENABLE SINGLE BUS CYCLE CLOCK MODE
'SET STEP STATE'                  -ENABLE SINGLE TIME STATE CLOCK MODE
'SET STEP INSTRUCTION' -ENABLES SINGLE INSTRUCTION MODE
'CLEAR STEP'                      -ENABLE NORMAL(NO STEP) MODE
'NEXT <NUMBER>'                   -<NUMBER> STEP CYCLES ARE DONE, TYPE OF STEP
                                   DEPENDS ON LAST 'SET STEP' COMMAND
'QCLEAR <ADDRESS>'                -DOES A QUAD CLEAR TO <ADDRESS>,WHICH IS FORCED
                                   TO A QUAD WORD BOUNDARY(CLEARS ECC ERRORS)
'SET SOMM'                        -SETS 'STOP ON MICRO-BREAK MATCH' ENABLE
'CLEAR SOMM'                      -CLEARS 'STOP ON MICRO-BREAK MATCH' ENABLE
'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 RELOCATION:<NUMBER>'         -PUTS <NUMBER> INTO THE CONSOLE'S RELOCATION
                                   REGISTER. CONTENTS OF RELOCATION REGISTER
                                   ARE ADDED TO EFFECTIVE ADDRESS OF PHYSICAL
                                   AND VIRTUAL EXAMINES AND DEPOSITS.
'SET DEFAULT <OPTION>,...,<OPTION>'   -SET CONSOLE DEFAULTS
      NOTE: <OPTIONS> ARE:        OCTAL,HEX,PHYSICAL,VIRTUAL,INTERNAL
                                  GENERAL,VBUS,IDBUS,BYTE,WORD,LONG,QUAD
'SET TERMINAL FILL:<NUMBER>'      -SETS FILL COUNT FOR NUMBER OF BLANKS WRITTEN
                                   TO THE TERMINAL AFTER <CR> OR <LF>
'SET TERMINAL PROGRAM' -PUTS CONSOLE TERMINAL INTO 'PROGRAM I/O' MODE
'^P'(CONTROL-P)                   -CAUSES CONSOLE TO EXIT 'PROGRAM I/O' MODE,
                                  -UNLESS MODE SWITCH IN A 'DISABLE' POSITION
'HELP'                    -PRINTS THIS FILE
'@<FILENAME>'                     -PROCESS AN INDIRECT COMMAND FILE
'LOAD <FILENAME>'                 -LOAD FILE TO MAIN MEMORY.
'LOAD/WCS <FILENAME>'             -LOAD FILE SPECIFIED TO WCS
      NOTE:   THE '/START:<ADDRESS>' QUALIFIER MAY ALSO BE USED TO SPECIFY
              THE STARTING ADDRESS FOR A LOAD, OTHERWISE LOAD WILL BEGIN
              WITH LOCATION 0.
```

CONSOLE HELP FILE

'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)

'PERFORM' -EXECUTE A FILE OF LINKED COMMANDS PREVIOUSLY GENERATED VIA A 'LINK' COMMAND.

'REPEAT <ANY-CONSOLE-COMMAND>' - CAUSES THE CONSOLE TO REPEATEDLY EXECUTE THE <CONSOLE-COMMAND>, UNTIL STOPPED BY A ^C

'WCS' -CALLS MICRO-DEBUGGER. (FOR DEBUGGER HELP, TYPE '@WCSMON.HLP')

'ENABLE DX1:' -ENABLES CONSOLE SOFTWARE TO ACCESS FLOPPY DRIVE 1 ON THOSE SYSTEMS WITH DUAL FLOPPIES.

'REBOOT' -CAUSES A CONSOLE SOFTWARE RELOAD

'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-11/780(COMMAND FILE EXECUTION WILL CONTINUE), OR B) THE VAX-11/780 HALTS, OR OPERATOR TYPES A ^C(COMMAND FILE EXECUTION WILL TERMINATE).

CONSOLE ABBREVIATION RULES

```
VAX-11/780 CONSOLE ABBREVIATION RULES   REV=5 8-NOV-77
 THIS FILE SHOWS THE SHORTEST UNIQUE COMMAND STRINGS THAT WILL BE
 RECOGNIZED AS THE CONSOLE COMMAND LISTED.

       COMMAND                    SHORTEST ABBREVIATION RECOGNIZED
 'EXAMINE <ADDRESS>'              'E <ADDRESS>'
 'DEPOSIT <ADDRESS> <DATA>'       'D <ADDRESS> <DATA>'
 'START <ADDRESS>'                'S <ADDRESS>'
 'CONTINUE'                       'C'
 'HALT'                  'H'
 'HELP'                  'HE'
 'BOOT'                  'B'
 'INITIALIZE'                     'I'
 'SHOW'                  'SH'
 'SHOW VERSION'          'SH V'
 'TEST'                  'T'
 'UNJAM'                          'U'
 'SET RELOCATION:<NUMBER>'        'SE R:<NUMBER>'
 'SET STEP BUS'          'SE S B'
 'SET STEP STATE'                 'SE S S'
 'SET STEP INSTRUCTION' 'SE S I'
 'CLEAR STEP'                     'CL S'
 'NEXT <NUMBER>'                  'N <NUMBER>'
 'QCLEAR <ADDRESS>'               'Q <ADDRESS>'
 'SET SOMM'                       'SE SO'
 'CLEAR SOMM'                     'CL SO'
 'SET CLOCK FAST'                 'SE C F'
 'SET CLOCK SLOW'                 'SE C S'
 'SET CLOCK NORMAL'               'SE C N'
 '@<FILENAME>'                    '@<FILENAME>'
```

CONSOLE ABBREVIATION RULES

'LOAD <FILENAME>'	'L <FILENAME>'
'LINK'	'LI'
'PERFORM'	'P'
'REPEAT <CONSOLE-COMMAND>'	'R <CONSOLE-COMMAND>'
'WCS'	'W'
'ENABLE DX1:'	'EN DX1:'
'REBOOT'	'REB'
'SET TERMINAL FILL:<NUMBER>'	'SE T F:<NUMBER>'
'SET TERMINAL PROGRAM'	'SE T PR'
'SET DEFAULT <OPTION-LIST>'	'SE D <OPTION-LIST>'
'WAIT DONE'	'WA D'

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>

CONSOLE-REMOTE ACCESS HELP FILE

'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 THE LOCAL TERMINAL TO CONTROL THE SYSTEM WHEN THE CONSOLE SWITCH IS IN THE REMOTE POSITIONS. DISABLED BY A CONTROL-P FROM THE REMOTE TERMINAL.

'ENABLE CARRIER ERROR'-CAUSES THE CONSOLE TO PRINT '?CARRIER LOST' WHEN A LOSS OF CARRIER IS DETECTED AT REMOTE INTERFACE.

'DISABLE ECHO' -INHIBITS ECHOING 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.

MICRODEBUGGER HELP FILE

```
                                REV-0   MAY 1977
!TO STOP PRINTING, TYPE ^C
>
 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,S1,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.
```

MICRODEBUGGER HELP FILE

```
'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.

'RETURN'                         -RETURN TO THE CONSOLE PROGRAM

'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.
```

ERROR MESSAGE HELP FILE

```
?'<TEXT-STRING>' IS INCOMPLETE
      THE <TEXT-STRING> IS NOT A COMPLETE CONSOLE COMMAND
?'<TEXT-STRING>' IS INCORRECT
      THE <TEXT-STRING> IS NOT RECOGNIZED AS PART OF A COMMAND
?FILE NAME ERR
      A <FILE-NAME> GIVEN WITH A COMMAND CAN NOT BE TRANSLATED TO RAD50
?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
?CPU NOT IN CONSOLE WAIT LOOP, COMMAND ABORTED
      A CONSOLE COMMAND REQUIRING ASSISTANCE FROM CPU WAS ISSUED
      WHILE THE CPU WAS NOT IN THE CONSOLE SERVICE LOOP
?CPU CLK STOP, COMMAND ABORTED
      A CONSOLE COMMAND THAT REQUIRES THE CPU CLOCK TO BE RUNNING
      WAS ISSUED WHILE THE CLOCK WAS STOPPED
?FLOPPY ERR,CODE=X
      THE CONSOLE FLOPPY DRIVER DETECTED AN ERROR. CODES ARE AS
      FOLLOWS:(CODES ARE ALWAYS PRINTED IN HEX RADIX)
      CODE=1  FLOPPY HARDWARE ERROR(CRC,PARITY,ETC)
      CODE=2  FILE NOT FOUND
      CODE=3  FLOPPY DRIVER QUEUE OVERFLOW
      CODE=4  CONSOLE SOFTWARE REQUESTED AN ILLEGAL SECTOR NUMBER
?FLOPPY NOT READY
      THE CONSOLE FLOPPY DRIVE FAILED TO BECOME READY WHEN BOOTING.
?NO BOOT ON FLOPPY
      CONSOLE ATTEMPTED TO BOOT FROM A FLOPPY THAT DOES NOT CONTAIN
      A VALID BOOT BLOCK.
?FLOPPY ERROR ON BOOT
      A FLOPPY ERROR WAS DETECTED WHILE ATTEMPTING A CONSOLE BOOT.
```

?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.

?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.

?INT-STK INVLD

THE CPU HALTED BECAUSE THE INTERRUPT STACK WAS MARKED INVALID

?CPU DBLE-ERR HLT

THE CPU HAS DONE A 'DOUBLE ERROR' HALT

?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

?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

ERROR MESSAGE HELP FILE

?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, 2) A COMMAND LINE DID NOT END WITH A CARRIGE RETURN AND LINE FEED.

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.

?WARNING-WCS & FPLA VER MISMATCH

THE MICROCODE IN WCS IS NOT COMPATIBLE WITH 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.

?MICRO-MACHINE TIME OUT

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

?REMOTE ACCESS NOT SUPPORTED

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

?TRAP-4, RESTARTING CONSOLE

THE CONSOLE TOOK A TIME-OUT TRAP. CONSOLE WILL RESTART.

?UNEXPECTED TRAP

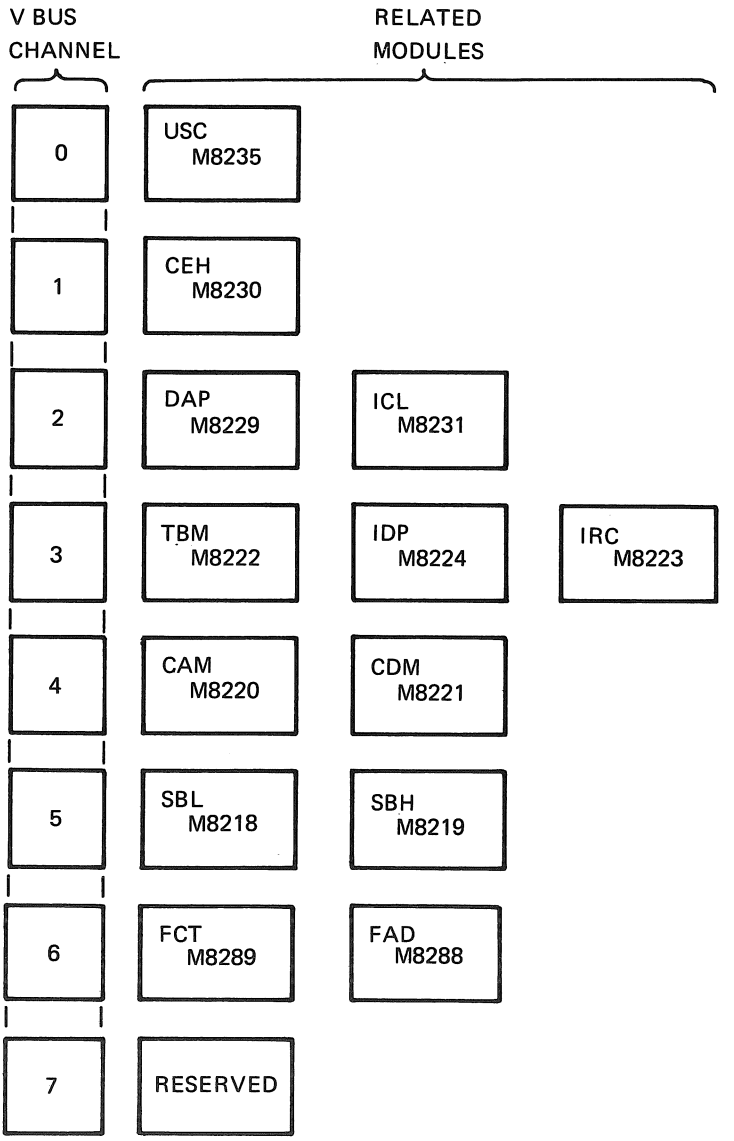
MOUNT CONSOLE FLOPPY, THEN TYPE ^C

CONSOLE TRAPPED TO AN UNUSED VECTOR. CONSOLE REBOOTS WHEN ^C TYPED

?Q-BLKD'

CONSOLE'S TERMINAL OUTPUT QUEUE IS BLOCKED. CONSOLE WILL REBOOT.

V BUS CHANNEL CONFIGURATION



TK-0703

V BUS DIRECTORY

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
00	0000	00000	USCF	M8235	CPT0	USCF UPCSV 00 H
00	0001	00001	USCF	M8235	CPT0	USCF UPCSV 01 H
00	0002	00002	USCF	M8235	CPT0	USCF UPCSV 03 H
00	0003	00003	USCF	M8235	CPT0	USCF UPCSV 03 H
00	0004	00004	USCF	M8235	CPT0	USCF UPCSV 04 H
00	0005	00005	USCF	M8235	CPT0	USCF UPCSV 05 H
00	0006	00006	USCF	M8235	CPT0	USCF UPCSV 06 H
00	0007	00007	USCF	M8235	CPT0	USCF UPCSV 07 H
00	0008	00010	USCF	M8235	CPT0	USCF UPCSV 08 H
00	0009	00011	USCF	M8235	CPT0	USCF UPCSV 09 H
00	000A	00012	USCF	M8235	CPT0	USCF UPCSV 10 H
00	000B	00013	USCF	M8235	CPT0	USCF UPCSV 11 H
00	000C	00014	USCF	M8235	CPT0	USCF UPCSV 12 H
00	000D	00015	USCB	M8235	CPTX	USCB STALL H
00	000E	00016	USCB	M8235	CPTX	USCB UTRAP H
00	000F	00017	USCJ	M8235	CPTX	USCJ ECO DISPATCH 06 H
00	0010	00020	USCE	M8235	CPT1	USCE ID BUS XCVR EN L
00	0011	00021	USCE	M8235	CPT3	USCE CS WR (31:00) H
00	0012	00022	USCE	M8235	CPT3	USCE CS WR (63:32) H
00	0013	00023	USCE	M8235	CPT3	USCE CS WR (95:64) H
00	0014	00024	USCE	M8235	CPTX	USCE WCS WR CYCLE H
00	0015	00025	USCE	M8235	CPT1	USCE WCS MEM AVAIL L
00	0016	00026	USCL	M8235	CPTX	FCTX ACC OVERRIDE L
00	0017	00027	USCM	M8235	CPTX	USCM IBUF EN (07:00) L
00	0018	00030	USCN	M8235	CPTX	A
00	0019	00031	USCN	M8235	CPTX	ICLK ALU Z (1) H
00	001A	00032	USCN	M8235	CPTX	DCPA LA00 H
00	001B	00033	USCN	M8235	CPTX	D
00	001C	00034	USCN	M8235	CPTX	E
00	001D	00035	USCN	M8235	CPTX	F
00	001E	00036	USCN	M8235	CPTX	ACCA UB0 H
00	001F	00037	USCN	M8235	CPTX	J
00	0020	00040	USCN	M8235	CPTX	K
00	0021	00041	USCN	M8235	CPTX	CEHB PSL C BIT H
00	0022	00042	USCN	M8235	CPTX	ICLK ALU C (1) H
00	0023	00043	USCN	M8235	CPTX	N
00	0024	00044	USCN	M8235	CPTX	P
00	0025	00045	USCN	M8235	CPTX	ACCA UB1 H
00	0026	00046	USCN	M8235	CPTX	S
00	0027	00047	USCN	M8235	CPTX	T
00	0028	00050	USCN	M8235	CPTX	DCPA LA01 H
00	0029	00051	USCN	M8235	CPTX	V
00	002A	00052	USCN	M8235	CPTX	X
00	002B	00053	USCN	M8235	CPTX	Y
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

V BUS DIRECTORY

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
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
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

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
01	0000	00000	CEHE	M8230	CPTX	PCSC PAR ERR (95:64) H
01	0001	00001	CEHE	M8230	CPTX	PCSC PAR ERR (63:32) H
01	0002	00002	CEHE	M8230	CPTX	PCSC PAR ERR (31:00) H
01	0003	00003	CEHE	M8230	CPTX	SBLP PAR ERR TRAP L
01	0004	00004	CEHE	M8230	CPTX	TBMW PROT UTRAP L
01	0005	00005	CEHF	M8230	CPTX	CEHF IRD STATE H
01	0006	00006	CEHF	M8230	CPTX	CEHF READ RLOG H
01	0007	00007	CEHF	M8230	CPTX	CEHF LOAD STATE H
01	0008	00010	CEHF	M8230	CPTX	CEHF CLR UWORD (1) H
01	0009	00011	CEHP	M8230	CPTX	ICLS EN ID XCEIV L
01	000A	00012	CEHA	M8230	CPTX	DEPM BMX31 L
01	000B	00013	CEHA	M8230	CPTX	DDPD BMX15 L
01	000C	00014	CEHA	M8230	CPTX	DCPD BMX07 L
01	000D	00015	CEHA	M8230	CPTX	DEPD AMX31 L
01	000E	00016	CEHA	M8230	CPTX	DDPB AMX15 L
01	000F	00017	CEHA	M8230	CPTX	DCPD AMX07 L
01	0010	00020	CEHA	M8230	CPTX	DEPK ALU BUFF31 L
01	0011	00021	CEHA	M8230	CPTX	DCPL ALU CARRY31 L
01	0012	00022	CEHA	M8230	CPTX	DCPL ALU CARRY15 L
01	0013	00023	CEHA	M8230	CPTX	DCPL ALU CARRY07 L
01	0014	00024	CEHA	M8230	CPTX	CEHA ALU BUFF17 L
01	0015	00025	CEHA	M8230	CPTX	CEHA ALU BUFF16 L
01	0016	00026	CEHA	M8230	CPTX	CEHA ALU BUFF15 L
01	0017	00027	CEHA	M8230	CPTX	CEHA ALU BUFF07 L
01	0018	00030	CEHA	M8230	CPTX	DEPN ALU(30:16)=0 L
01	0019	00031	CEHA	M8230	CPTX	DDPF ALU(15:08)=0 L
01	001A	00032	CEHA	M8230	CPTX	DCPF ALU(07:00)=0 L
01	001B	00033	CEHA	M8230	CPTX	BUS ALU BYTE2,3 A=B H
01	001C	00034	CEHA	M8230	CPTX	BUS ALU BYTE1 A=B H
01	001D	00035	CEHA	M8230	CPTX	BUS ALU BYTE0 A=B H
01	001E	00036	CEHB	M8230	CPTX	DCPA AMX00 L
01	001F	00037	CEHB	M8230	CPTX	DAPB AUALU=A PLUS B L
01	0020	00040	CEHB	M8230	CPTX	DAPB AUALU=A MINUS B L
01	0021	00041	CEHC	M8230	CPTX	DDPN EALU09 H
01	0022	00042	CEHC	M8230	CPTX	DDPN EALU08 H
01	0023	00043	CEHC	M8230	CPTX	ACCX NDATA H
01	0024	00044	CEHC	M8230	CPTX	ACCX ZDATA H
01	0025	00045	CEHC	M8230	CPTX	ACCX VDATA H
01	0026	00046	CEHC	M8230	CPTX	ACCX CDATA H
01	0027	00047	CEHD	M8230	CPTX	CEHD SECOND REF H
01	0028	00050	CEHD	M8230	CPTX	SBLT STALL L
01	0029	00051	CEHD	M8230	CPTX	IRCJ DQ CONT H
01	002A	00052	CEHD	M8230	CPTX	IRCJ FLOAT H
01	002B	00053	CEHD	M8230	CPTX	IRCJ WORD CONT H
01	002C	00054	CEHD	M8230	CPTX	IRCJ BYTE CONT H
01	002D	00055	CEHD	M8230	CPTX	TBMW SAVE CONTEXT H
01	002E	00056	CEHE	M8230	CPTX	DDPS FLOAT NZERO H
01	002F	00057	CEHE	M8230	CPTX	USCB CLR UTRAP L
01	0030	00060	CEHR	M8230	CPTX	DCPH VA02(1) H
01	0031	00061	CEHR	M8230	CPTX	DCPJ VA01(1) H
01	0032	00062	CEHR	M8230	CPTX	DCPJ VA00(1) H
01	0033	00063	CEHE	M8230	CPTX	TBMW EN CMODADRS H
01	0034	00064	CEHE	M8230	CPTX	TBMN PAGE EDGE H
01	0035	00065	CEHE	M8230	CPTX	TBMW EN UNALIGN TRAP H
01	0036	00066	CEHE	M8230	CPTX	SBLM TIMEOUT TRAP L
01	0037	00067	CEHE	M8230	CPTX	SBLR RDS TRAP L
01	0038	00070	CEHE	M8230	CPTX	TBMW TB PAR UTRAP L
01	0039	00071	CEHE	M8230	CPTX	TBMW MISS UTRAP L
01	003A	00072	CEHE	M8230	CPTX	TBMW MBIT UTRAP L
01	003B	00073	CEHE	M8230	CPTX	CEHE CS PE TRAP H
01	003C	00074	CEHX	M8230	CPTX	RESERVED
01	003D	00075	CEHX	M8230	CPTX	RESERVED
01	003E	00076	CEHX	M8230	CPTX	RESERVED
01	003F	00077	CEHX	M8230	CPTX	RESERVED

V BUS DIRECTORY

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	000B	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
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

V BUS DIRECTORY

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
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	0048	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
02	0054	00124	ICLE	M8231	CPTX	ICLE REM BEMX S0 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	CPTX	ICLH ID TO ATMP L
02	0059	00131	ICLH	M8231	CPTX	ICLH ID TO BTMP L
02	005A	00132	ICLH	M8231	CPTX	ICLH IDM S2 L
02	005B	00133	ICLH	M8231	CPTX	ICLH IDM S1 L
02	005C	00134	ICLH	M8231	CPTX	ICLH IDM S0 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 0 L
02	0065	00145	ICLK	M8231	CPTX	DDPN EALU09 H
02	0066	00146	ICLK	M8231	CPTX	DDPN EALU=0 L
02	0067	00147	ICLX	M8231	CPTX	RESERVED

V BUS DIRECTORY

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	CPT0	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	USCB 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	M8224	CPT0	IDPA BUF B0-7(1) H
03	0021	00041	IDPA	M8224	CPT0	IDPA BUF B0-6(1) H
03	0022	00042	IDPA	M8224	CPT0	IDPA BUF B0-5(1) H
03	0023	00043	IDPA	M8224	CPT0	IDPA BUF B0-4(1) H
03	0024	00044	IDPA	M8224	CPT0	IDPA BUF B0-3(1) H
03	0025	00045	IDPA	M8224	CPT0	IDPA BUF B0-2(1) H
03	0026	00046	IDPA	M8224	CPT0	IDPA BUF B0-1(1) H
03	0027	00047	IDPA	M8224	CPT0	IDPA BUF B0-0(1) H
03	0028	00050	IDPA	M8224	CPT0	IDPA BUF B1-7(1) H
03	0029	00051	IDPA	M8224	CPT0	IDPA BUF B1-6(1) H
03	002A	00052	IDPA	M8224	CPT0	IDPA BUF B1-5(1) H
03	002B	00053	IDPA	M8224	CPT0	IDPA BUF B1-4(1) H
03	002C	00054	IDPA	M8224	CPT0	IDPA BUF B1-3(1) H
03	002D	00055	IDPA	M8224	CPT0	IDPA BUF B1-2(1) H
03	002E	00056	IDPA	M8224	CPT0	IDPA BUF B1-1(1) H
03	002F	00057	IDPA	M8224	CPT0	IDPA BUF B1-0(1) H
03	0030	00060	IDPH	M8224	CPT0	IDPH IBC 3(1) H
03	0031	00061	IDPH	M8224	CPT0	IDPH IBC 2(1) H
03	0032	00062	IDPH	M8224	CPT0	IDPH IBC 1(1) H
03	0033	00063	IDPH	M8224	CPT0	IDPH IBC 0(1) H
03	0034	00064	IDPH	M8224	CPTX	IDPH CLR 0 L

V BUS DIRECTORY

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
03	0035	00065	IDPH	M8224	CPTX	IRCE SAVE H
03	0036	00066	IDPA	M8224	CPT0	IDPA VAX H
03	0037	00067	IDPA	M8224	CPTX	IDPA DST R MODE H
03	0038	00070	IDPJ	M8224	CPTX	SBLR IB READ DATA L
03	0039	00071	IDPX	M8224	CPTX	RESERVED
03	003A	00072	IDPJ	M8224	CPT0	IDPJ COUNT H
03	003B	00073	IDPJ	M8224	CPTX	IDPJ FLUSH L
03	003C	00074	IDPJ	M8224	CPTX	IDPJ B5 VAL(1) H
03	003D	00075	IDPJ	M8224	CPTX	IDPJ B4 VAL(1) H
03	003E	00076	IDPJ	M8224	CPTX	IDPJ B3 VAL(0) H
03	003F	00077	IDPJ	M8224	CPTX	IDPJ B2 VAL(0) H
03	0040	00100	IDPM	M8224	CPTX	IRCD PC MODE H
03	0041	00101	IDPM	M8224	CPTX	IRCD SEL LONG L
03	0042	00102	IDPM	M8224	CPTX	IRCD SEL WORD L
03	0043	00103	IDPM	M8224	CPTX	IRCD SEL BYTE H
03	0044	00104	IDPM	M8224	CPTX	IRCE CTX 3 L
03	0045	00105	IDPM	M8224	CPTX	IRCE CTX 2 L
03	0046	00106	IDPL	M8224	CPTX	IDPL ID BUS XCVR EN L
03	0047	00107	IDPX	M8224	CPTX	RESERVED
03	0048	00110	IDPM	M8224	CPTX	IDPM B DELTA PC 2 H
03	0049	00111	IDPM	M8224	CPTX	IDPM 16 BIT B DEST L
03	004A	00112	IDPM	M8224	CPTX	IDPM B DEST H
03	004B	00113	IDPM	M8224	CPTX	IDPM B DELTA PC 1 H
03	004C	00114	IDPM	M8224	CPTX	IDPM VAXSL L
03	004D	00115	IDPM	M8224	CPTX	IDPM B DELTA PC 0 H
03	004E	00116	IDPM	M8224	CPTX	TBMX VA 01 H
03	004F	00117	IDPM	M8224	CPTX	TBMX VA 00 H
03	0050	00120	IRCH	M8223	CPTX	TBMX IB ERR L
03	0051	00121	IRCH	M8223	CPTX	TBMX TB MISS L
03	0052	00122	IRCC	M8223	CPTX	CEHH FPD BIT L
03	0053	00123	IRCX	M8223	CPTX	RESERVED
03	0054	00124	IRCE	M8223	CPTX	IRCE IB ADVANCE H
03	0055	00125	IRCJ	M8223	CPTX	IRCJ SP2 CON 1 H
03	0056	00126	IRCJ	M8223	CPTX	IRCJ SP2 CON 0 H
03	0057	00127	IRCM	M8223	CPTX	IRCM DATA EN L
03	0058	00130	IRCE	M8223	CPT0	ICLD SERVICE H
03	0059	00131	IRCE	M8223	CPT0	ICLD SERVICE BIT 0 H
03	005A	00132	IRCE	M8223	CPT0	ICLD SERVICE BIT 1 H
03	005B	00133	IRCE	M8223	CPT0	ICLD SERVICE BIT 2 H
03	005C	00134	IRCC	M8223	CPT0	IRCC EXEC CT 0 H
03	005D	00135	IRCC	M8223	CPT0	IRCC EXEC CT 1 H
03	005E	00136	IRCC	M8223	CPT0	IRCC EXEC CT 2 H
03	005F	00137	IRCX	M8223	CPT0	RESERVED

V BUS DIRECTORY

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	000B	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	001B	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
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

V BUS DIRECTORY

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
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
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	005B	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

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
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

V BUS DIRECTORY

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

V BUS DIRECTORY

CHAN	BIT (HEX)	BIT (OCTAL)	DWG	MODULE	T.S.	SIGNAL NAME
06	0000	00000	FCTP	M8288	CPTX	DAPL ACC RA CONTEXT 0 H
06	0001	00001	FCTP	M8288	CPTX	DAPL ACC RA CONTEXT 1 H
06	0002	00002	FCTC	M8288	CPTX	FCTC CLR RR L
06	0003	00003	FCTH	M8288	CPTX	FCTH CP SYNC H
06	0004	00004	FNME	M8288	CPTX	FNME BUS_EXP L
06	0005	00005	FCTJ	M8288	CPTX	FCTJ ACC N DATA H
06	0006	00006	FCTC	M8288	CPTX	FCTC ACC Z DATA H
06	0007	00007	FCTC	M8288	CPTX	FCTC ACC V DATA H
06	0008	00010	FNMT	M8288	CPT3	FCTD RA ADRS 3 L
06	0009	00011	FNMT	M8288	CPT3	FCTD RA ADRS 2 L
06	000A	00012	FNMT	M8288	CPT3	FCTD RA ADRS 1 L
06	000B	00013	FNMT	M8288	CPT3	FCTD RA ADRS 0 L
06	000C	00014	FNMT	M8288	CPT3	FCTP RB ADRS 3 L
06	000D	00015	FNMT	M8288	CPT3	FCTP RB ADRS 2 L
06	000E	00016	FNMT	M8288	CPT3	FCTP RB ADRS 1 L
06	000F	00017	FNMT	M8288	CPT3	FCTP RB ADRS 0 L
06	0010	00020	XXXX	M8288	CPTX	RESERVED
06	0011	00021	XXXX	M8288	CPTX	RESERVED
06	0012	00022	FNMT	M8288	CPTX	EALU C 0 L
06	0013	00023	FNMT	M8288	CPTX	FCTE COMPL L
06	0014	00024	FNMT	M8288	CPTX	FADA SPC (0) H
06	0015	00025	FNMT	M8288	CPTX	FNMB EALU CIN L
06	0016	00026	FNMT	M8288	CPTX	FCTC SEL NORM H
06	0017	00027	FNMT	M8288	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
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

EXPLANATION OF VERSION NUMBERS FOR CONSOLE BOOTING

VER PCS=01 WCS=03-10 FPLA=03 CON PX03-08

I. WHERE VERSION NUMBERS COME FROM
   A. PCS (i.e., 01 in example)
      1. Version stored in CID field on location 111 in PCS
   B. WCS
      1. Primary version number (i.e., 03 in example)
         a. Version stored in CID field of location 1111 in WCS
      2. Secondary version number (i.e., 10 in example)
         a. Stored in CID field of location 1112 in WCS
   C. How read
      The console puts the microaddress on the microstack, asserts ROM NOP, does a maintenance return, steps the microcode to the proper place, and then reads the ID address bits (that come from the control store ROM) from bits <12:8> of the ID C/S register on CIB. The console software then stores the complement of these bits in temporaries in 1103 memory.
   D. FPLA (03 in example)
      The console initiates a maintenance return from location "F80" which is an "ECO" location in the FPLA. The console steps the microcode to the location following "F80" and reads the Jump field over the V-Bus, the least significant six bits of the jump field are the version number of the FPLA.
   E. CON (PX03-08 in example)
      1. Version of console software read from 1103 memory.

II. ERROR MESSAGES
   A. "Warning-WCS & FPLA version mismatch"
      1. Checks WCS primary version and FPLA versions are the same.
   B. "Fatal-WCS & PCS version mismatch"
      1. Check WCS secondary version and PCS version is the same. Bits <5:4> of the WCS secondary version are compared to Bits <1:0> of the PCS version.

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

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

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

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
F4 - CP Read Timeout or Error Confirmation Abort
F5 - CP Read Data Substitute Abort
F6 - Microcode "not supposed to get here" abort

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

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

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

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

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

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

MICRODIAGNOSTICS RUN, CONSOLE TERMINAL OUTPUT

```
>>>H
        HALTED AT 800082EE
>>>I
        INIT SEQ DONE
>>>U

>>>TEST                                  ;Console prompt, test command

ZZ-ESKAB    V9.0                        ;Program title and version
01,02,03,04,                            ;Section numbers
NO. OF WCS MODULES = 0002               ;Configuration information
05,06,07,08,09,0A,0B,0C,0D,0E,0F,10,11,12,13,14,15,16,17,
18,19,1A,1B,1C,1D,1E,1F,20,21,22,23,24,25,26,27,28,29,2A,2B,2C,2D,2E,
2F,30,31,32,33,34,35,36,
END PASS 0001
MOUNT FLOPPY ZZ-ESZAD & TYPE "DI"       ;instructions for second half
                                        ;microdiagnostic monitor prompt
MIC>DI
37,38,39,3A,3B,3C,3D,3E,                ;section number(s)
CPU TR= 00000010                        ;configuration information
MS780 4K CHIP AT TR 01                  ;which is system specific
MAX ADR+1= 00090000
DW780 AT TR 03
RH780 AT TR 08
RH780 AT TR 09
3F,40,41,42,43,44,45,46,47,48,49,4A,4B,4C,4D,4E,4F,50,
STARTING FPA TESTS
51,52,53,54,55,56,57,58,59,5A,5B,5C,5D,
END PASS 0001

        CPU HALTED,SOMM CLEAR,STEP=NONE,CLOCK=NORM      ;CPU status
        RAD=HEX,ADD=PHYS,DAT=LONG,FILL=00,REL=00000000  ;and console
        INIT SEQ DONE                                   ;program
        HALTED AT 00000000                              ;defaults

        (RELOADING WCS)                                 ;configuration
        LOAD DONE, 00003200 BYTES LOADED                ;information
        VER: PCS=01 WCS=08-11 FPLA=08 CON=V02-03-L      ;console prompt
>>>
```

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

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

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

HALT       ! HALT THE CPU   ; These commands
                            ; and responses are
           CPU HALTED       ; produced automatically
INIT       ! INITIALIZE     ; down to <@EXIT>

           INIT SEQ DONE
UNJAM      ! AND CLEAR THE SBI

LO ESKAX.EXE/ST:200

           LOAD DONE, 0000DC00 BYTES LOADED
!          QUICK VERIFY

           LOAD DONE, 0000D000 BYTES LOADED
START 10000

<@EOF>
<@EXIT>

DIAGNOSTIC SUPERVISOR. ZZ-ESSAA-4.02.417 6-JUN-1978 08:39:21.31

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

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

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

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

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

NOTE

Operator input is underlined.

LOAD AND RUN MACRODIAGNOSTICS UNDER VMS (ON-LINE)

```
^Y                          ; Return control to VMS.

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

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

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

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

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 micro-diagnostics the specified number of passes before returning to the console. If the number is -1, execute the micro-diagnostics 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 0 is at the right side of the display.
EXAMINE RA:<ADDRESS>	Displays the contents of the RA Scratch Pad specified by <ADDRESS>.
EXAMINE RC:<ADDRESS>	Displays the contents of the RC Scratch Pad specified by <ADDRESS>.
EXAMINE LA	Displays the contents of the LA Latch.
EXAMINE LC	Displays the contents of the LC Latch.
EXAMINE DR	Displays the contents of the D Register.
EXAMINE QR	Displays the contents of the Q Register.
EXAMINE SC	Displays the contents of the SC Register.
EXAMINE 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>

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>.

MICRODIAGNOSTIC PSEUDO INSTRUCTION DEFINITIONS

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.0XT=3                      ;RAMX ZERO EXTENDED. 0XT(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) *, G 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=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/=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 UKMX
        WRITE.SC=0D                     ;WRITE REG SELECTED BY SC
        WRITE.KMX=0F                    ;WRITE REG SELECTED BY UKMX

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.WCHK=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.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/=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.OXT,DT/LONG,ALU/A"
ALU_0[]D        "ALU/@1,AMX/RAMX.OXT,LONG,BMX/RBMX,RBMX/D"
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+D         "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A+B"
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+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          "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-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           "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+Q+1         "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B+1"
ALU_-1            "AMX/RAMX.OXT,DT/LONG,ALU/NOTA"
ALU_D             "RAMX/D,AMX/RAMX,ALU/A"
ALU_D.OXT[]       "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A"
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[]+K[]  "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
ALU_D.OXT[]-K[]  "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B"
ALU_D.OXT[]+LC   "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/LC"
ALU_D.OXT[]+Q    "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/RBMX,RBMX/Q"
ALU_D.OXT[]-Q    "RAMX/D,AMX/RAMX.OXT,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.OXT,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.OXT[]        "RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A"
ALU_Q.OXT[].ANDNOT.K[]   "ALU/ANDNOT,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.OXT[].OR.K[]       "ALU/OR,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.OXT[].OR.D         "ALU/OR,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/RBMX,RBMX/D"
ALU_Q.OXT[]+D      "ALU/A+B,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RBMX/D,RAMX/Q"
ALU_Q.OXT[]+D+1    "ALU/A+B+1,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RAMX/Q,RBMX/D"
ALU_Q.OXT[]+K[]    "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
ALU_Q.OXT[]-K[]    "ALU/A-B,AMX/RAMX.OXT,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.R/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.OXT,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.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-K[]          "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_0-LC           "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_0+MASK+1       "AMX/RAMX.OXT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,QK/SHF"
Q_0+PC.RLOG      "AMX/RAMX.OXT,DT/LONG,BMX/PC,ALU/A+B.RLOG,SHF/ALU,QK/SHF"
Q_0-Q            "AMX/RAMX.OXT,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.OXT[]        "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/ALU,QK/SHF"
Q_D.OXT[]+K[].LEFT "RAMX/D,AMX/RAMX.OXT,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.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[].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).AND.Q       "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.OR.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.RA3/@1,AMX/RAMX.OXT,DT/LONG,ALU/A,SHF/ALU"
R[]_0-D          "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-K[]        "AMX/RAMX.OXT,DT/LONG,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_0-LB         "AMX/RAMX.OXT,DT/LONG,BMX/LB,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB.SPO.RAB/@1"
```

```
R[]_D+LB+1          "AMX/RAMX.OXT,DT/LONG,BMX/LB,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_D-Q             "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
R[]_D-1             "AMX/RAMX.OXT,DT/LONG,BMX/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.O         "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... THRU R[]_RLOG...
R[]_Q             "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU"
R[]_Q+1           "ALU_O+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/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"
;RC[]_O... THRU RC[]_D...

RC[]_O            "AMX/RAMX.OXT,DT/LONG,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_O-D          "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_O+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[]_O+LC+1       "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_O+MASK+1     "AMX/RAMX.OXT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RC[]_O+MASK+1.RIGHT2      "AMX/RAMX.OXT,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.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(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.OXT,DT/LONG,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
SC_0-K[]          "SMX/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.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(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.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_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.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.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]"
;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,CLK.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"       ;,J5/1B"
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"       ;,U3/5" ;PREFERED FORM
EALU?             "BEN/EALU"
EALU.N?           "BEN/EALU"        ;,U4/07"
EALU.Z?           "BEN/EALU"        ;,U4/0B"
END.DP1?          "BEN/END.DP1"
FPD?              "BEN/LAST.REF"    ;,U4/07"
IB.TEST?          "BEN/IB.TEST"
INT?              "BEN/INTERRUPT"
INTERRUPT.REQ?    "BEN/INTERRUPT"   ;,U3/5"
IR0?              "BEN/ALU"         ;,U4/0D"
IR0.C31?          "BEN/ALU"
IR1?              "BEN/IR2-1"       ;,U3/6"
IR2-1?            "BEN/IR2-1"
LAST.REF?         "BEN/LAST.REF"
MODE.LSS.ASTLVL?          "BEN/REI"         :,U3/3"
MUL?              "BEN/MUL"
NEST.ERR?         "BEN/LAST.REF"    ;,U4/0B"
PC.MODES?         "BEN/PC.MODES"
PSL.C?            "BEN/PSL.CC"      ;,U4/0E"
PSL.CC?           "BEN/PSL.CC"
PSL.MODE?         "BEN/PSL.MODE"
PSL.N?            "BEN/PSL.CC"      :,U4/7"
PSL.V?            "BEN/PSL.CC"      ;,U4/0D"
PSL.Z?            "BEN/PSL.CC"      ;,U4/0B"
PTE.VALID?        "BEN/TB.TEST"     ;,U5/0F"
QUAD?             "BEN/DATA.TYPE"
Q31?              "BEN/SIGNS"       ;,U3/3"
RLOG.EMPTY?       "BEN/ALU1-0"      ;,U4/7"
ROR?              "BEN/ROR"
SC?               "BEN/SC"
SC.GT.0?          "BEN/SC"
SC.NE.0?          "BEN/MUL"         ;,U3/3"
SIGNS?            "BEN/SIGNS"
SRC.PC?           "BEN/SRC.PC"      ;COMP MODE, BEN ON SRC R = PC
SS?               "BEN/EALU"        ;,U4/0E"
STATE0?           "BEN/STATE3-0"    ;,U4/0E"
STATE1?           "BEN/STATE3-0"    ;,U4/0D"
STATE1-0?         "BEN/STATE3-0"    ;,U4/0C"
STATE2?           "BEN/STATE3-0"    ;,U4/0B"
```

```
STATE3?         "BEN/STATE3-0"   ;.U4/07"
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"   ;.U5/0F"
VA31-30?        "BEN/PSL.MODE"   ;.U5/07"
Z?              "BEN/Z"
ZONED?          "BEN/DECIMAL"    ;.U2/1"
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.OXT.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,RAMX/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.OXT,DT/LONG,BMX/MASK,ALU/A+B+1"
ALU_NOT.K[]     "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/ORNOT"
ALU_NOT.LA      "AMX/LA,ALU/NOTA"
ALU_NOT.MASK    "AMX/RAMX.OXT,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.OXT,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,SL/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.OXT,BMX/MASK,ALU/A+B+1,D_ALU"
D_NOT.RC[]        "AMX/RAMX.OXT,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 "SPD.R/LOAD.LAB,SPD.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)      "SPD.AC/LOAD.LAB,SPD.ACN/PRN"
LAB_R(PRN+1)    "SPD.AC/LOAD.LAB,SPD.ACN/PRN+1"
LAB_R(SP1+1)    "SPD.AC/LOAD.LAB,SPD.ACN/SP1+1"
LAB_R(SRC11)    "SPD.AC/LOAD.LAB,SPD.ACN11/SRC.OR.1"
LAB_R(SP2)      "SPD.AC/LOAD.LAB,SPD.ACN/SP2.SP2"
LAB_R(SP2.SP1)  "SPD.AC/LOAD.LAB,SPD.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,SPD.R/LOAD.LC,SPD.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.oxt,bmx/mask,alu/onot,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,SPD.R/LOAD.LAB,SPD.RAB/@1,AMX/LA,ALU/AND,Q_ALU"
Q_Q.ANDNOT.RC[] "RAMX/Q,AMX/RAMX,SPD.R/LOAD.LC,SPD.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[]_O-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.D        "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.RAB/@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"
```

DIAGNOSTIC SUPERVISOR COMMANDS

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

The diagnostic supervisor commands are grouped in three sets:

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

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

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

PROGRAM/TEST SEQUENCE CONTROL COMMANDS

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

Load Command

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

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

NOTE

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

Start Command

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

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

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

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

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

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

Restart Command

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

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

Run Command

RUN <file-spec> [/SEC: <section-name>]!
[/TEST:<first> [:<last>
!/SUBTECT:<num>]] [/PASS:<count>]

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

Control C

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

Continue Command

CONTINUE

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

Summary Command

SUMMARY

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

Abort Command

ABORT

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

EXECUTION CONTROL FUNCTIONS

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

Set Flags Command

SET [FLAGS] <arg-list>

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

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

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

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

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

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

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

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

QUICK — Quick verify. This flag, when set, indicates to the program that the operator desires a quick verify mode of operation. The interpretation of this flag is program dependent.

SPOOL — List error messages on line printer. This flag, when set, causes the supervisor to direct all program messages to the line printer. In the VMS environment, the messages are not actually printed but entered in a ile on disk.

TRACE — Report the execution of each test. When set, this flag causes the supervisor to report the execution of each individual test within the program as the supervisor dispatches control to that test.

LOCK — Lock in physical memory. When set, this flag disables the program relocation function. Self-relocating programs are then locked into their current physical memory space.

OPERATOR — Operator present. When set, this flag indicates to the supervisor that operator interaction is possible. When cleared, supervisor takes appropriate actions to insure that the test session continues without an operator.

PROMPT — Display long dialogue. When set, this flag indicates to the supervisor that the operator wants to see the limits and defaults for all questions printed by the program.

ALL All flags in this list.

Clear Flags Command

CLEAR [FLAGS] <arg-list>

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

Set Flags Default Command

SET FLAGS DEFAULT

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

Show Flags Command

SHOW FLAGS

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

Set Event Flags Command

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

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

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

Clear Event Flags Command

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

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

Show Event Flags Command

SHOW EVENT [FLAGS]

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

DEBUG AND UTILITY FEATURES

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

Set Base Command

SET BASE <virtual-address>

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

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

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

Set Breakpoint Command

SET BREAKPOINT <address>

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

Clear Breakpoint Command

CLEAR BREAKPOINT <address>!ALL

This command clears previously set breakpoints.

Show Breakpoints Command

SHOW BREAKPOINTS

This command displays all currently defined breakpoints.

Set Default Command

SET DEFAULT <arg-list>

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

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

Examine Command

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

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

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

Deposit Command

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

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

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

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

SECTION 7
SYSTEM OPERATION

```
        THIS FILE DESCRIBES THE INPUT PARAMETERS TO THE BOOTSTRAP PROGRAM
        VMB.EXE .  NORMALLY THE BOOTSTRAP WILL LOOKUP THE FILE [10,40]SYSBOOT.EXE
        ON THE SPECIFIED DEVICE, LOAD IT INTO MEMORY AND TRANSFER CONTROL
        TO IT.

        THE BOOTSTRAP IS LOADED INTO MEMORY AT LEAST ONE PAGE ABOVE THE FIRST
        AVAILABLE WORKING MEMORY TO ALLOW SPACE FOR THE RESTART PARAMETER
        BLOCK.  THE ADDRESS OF THE BASE OF THE BOOTSTRAP IS PASSED THROUGH
        SP, THE STACK POINTER, WHERE IT ALSO SERVES AS A TEMPORARY STACK POINTER.

INPUT PARAMETERS:
        R0  -  <31:4>=MBZ; <3:0>=DEVICE TYPE CODE
                        0 => DISK PACK          (RM03/RP04/RP05/RP06/RP07)
                        1 => CARTRIDGE DISK     (RK06/RK07)

        R1  -  <31:4>=MBZ; <3:0>=SYSTEM BUS ADDRESS("TR" NUMBER)
                FOR MOST CONFIGURATIONS THE FOLLOWING CONVENTION HAS BEEN
                USED:
                        TR NUMBER       ADAPTER / CONTROLLER
                        ---------       --------------------
                            3           UNIBUS ADAPTER
                            8           MASSBUS ADAPTER NUMBER 1
                            9           MASSBUS ADAPTER NUMBER 2

        R2  -  FOR UBA:
                        <31:18>=MBZ; <17:3>=UNIBUS ADDRESS OF CONTROL REGISTER
                        <2:0>=MBZ
                        RK06/RK07 CSR = 3FF20
               FOR MBA:
```

<31:4>=MBZ; <3:0>=CONTROLLER/FORMATTER NUMBER

R3 - <31:4>=MBZ; <3:0>=UNIT NUMBER

R4 - <31:0>=LOGICAL BLOCK NUMBER TO READ AS BOOT BLOCK

R5 - <31:0>=SOFTWARE BOOT CONTROL FLAGS

BIT	MEANING
0	CONVERSATIONAL BOOT. AT VARIOUS POINTS IN THE SYSTEM BOOT PROCEDURE, PARAMETER AND OTHER INPUT WILL BE SOLICITED FROM THE CONSOLE.
1	DEBUG. THIS FLAG IS PASSED THROUGH TO VMS AND CAUSES THE CODE FOR THE EXEC DEBUGGER TO BE INCLUDED IN THE RUNNING SYSTEM.
2	INITIAL BREAKPOINT. IF THIS FLAG IS SET, AND THE EXEC DEBUGGER CODE IS INCLUDED (FLAG BIT 1) THEN A BREAKPOINT WILL OCCUR IMMEDIATELY AFTER THE EXEC ENABLES MAPPING.
3	BOOT BLOCK. IF THIS FLAG IS SET THEN THE BOOT BLOCK WILL BE READ AND CONTROL TRANSFERED TO IT.
4	DIAGNOSTIC BOOT. THIS FLAG CAUSES A BOOT BY FILE NAME FOR THE DIAGNOSTIC SUPERVISOR.
5	BOOTSTRAP BREAKPOINT. THIS FLAG CAUSES THE BOOTSTRAP TO STOP A BREAKPOINT AFTER PERFORMING NECESSARY INITIALIZATION IF IT HAS BEEN BUILT WITH DEBUG CODE.
6	IMAGE HEADER. IF THIS FLAG IS SET THE TRANSFER ADDRESS FROM THE IMAGE HEADER OF THE BOOT FILE WILL BE USED.

OTHERWISE CONTROL WILL TRANSFER TO THE FIRST BYT OF THE BOOT FILE.

7 MEMORY TEST INHIBIT. THIS FLAG INHIBITS THE TESTING OF MEMORY DURING BOOTSTRAPPING.

8 FILE NAME. CAUSES THE BOOTSTRAP TO SOLICIT THE NAME OF THE BOOT FILE.

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

10 NO PFN DELETION. A FLAG PASSED THROUGH TO THE BOOTFILE THAT PREVENTS SPECIFIED PAGES FROM BEING PERMANENTLY REMOVED FROM THE POOL OF AVAILABLE PAGES.

R10 - HALT PC

R11 - HALT PSL

AP - HALT CODE

SP - ADDRESS+(^X200) OF FIRST WORKING 64KB MEMORY REGION

USABLE AS BOTH STACK POINTER AND POINTER TO GOOD MEMORY.

USE OF FILEX TO TRANSFER DIAGNOSTIC FILES

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

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

2. Type in the following commands under VMS.

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

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

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

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

FLX>/RS=DX1:<CODE>. EXE/RT/IM
                                ; Transfer the file from the floppy
                                ; disk to the system device, where
                                ; <CODE> is the mnemonic of the
                                ; program.  Transfer each file
                                ; needed in this manner.
```

```
FLX>/CO/BL:512./RS=DX1:ESSAA.EXE/RT/IM
                                ; transfer the diagnostic
                                ; supervisor.
FLX> ^Y                         ; Return control to VAX/VMS.

$ DISM DX1:                     ; Dismount the floppy disk.
$ DEALL DXA1:                   ; Deallocate the floppy disk.
$                               ; VAX/VMS                    prompt
```

3. Sample console terminal output:

```
$ MCR FLX
FLX> /CO/BL:512./RS=DX1:ESSAA.EXE/RT/IM
FLX> /RS=DX1:ESMAA.EXE/RT/IM
FLX> ^Y
$
```

TERMINAL FUNCTION KEYS

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

COMMANDS FOR TERMINAL COMMUNICATION AND CONTROL

:=	(String assignment.) Defines a local symbol name as a synonym for a DCL command.
:==	(String assignment.) Defines a global symbol name as a synonym for a DCL command.
HELP	Displays information about a command or a command parameter or qualifier on the terminal.
LOGOUT	Terminates communication between a user and the system.
MCR	Passes a command line to the RSX-11M MCR environment, or places the terminal in MCR command mode.
REQUEST	Displays a message at an operator's terminal.
SET PASSOWRD	Changes the password associated with your user name for subsequent logins.
SET TERMINAL	Defines the characteristics of the terminal for the duration of a terminal session.
SHOW DAYTIME	Displays the current data and time of day on the terminal.
SHOW SYMBOL	Displays current local or global symbols and the strings or values assigned to them.
SHOW TERMINAL	Displays the current characteristics of the terminal.

COMMANDS FOR FILE MANIPULATION

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

COMMANDS FOR FILE MANIPULATION

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

COMMANDS FOR DEVICE HANDLING

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

COMMANDS FOR PROGRAM DEVELOPMENT AND CONTROL

ASSIGN	Defines a file specification to be associated with a specific logical device name used in a program.
BASIC	Invokes the PDP-11 BASIC-PLUS-2/VAX compiler to compile BASIC language source statements.
CANCEL	Halts periodic execution of an image scheduled for execution in a process.
COBOL/RSX11	Invokes the PDP-11 COBOL-74/VAX compiler to compile a set of COBOL language source statements.
CONTINUE	Resumes execution of an interrupted command, program, or command procedure.
DEBUG	Invokes the VAX/VMS Symbolic Debugger to begin interactive debugging.
DEFINE	Equates character strings with logical names. These names can be accessed and translated from within user programs.
DEPOSIT	Replaces the contents of a location in virtual memory with new data or instruction.
EDIT	Invokes an editor to create or modify a source program or data file.
EXAMINE	Displays the contents of a location in virtual memory.
FORTRAN	Invokes the VAX-11 FORTRAN IV-PLUS compiler to compile a set of FORTRAN language source statements.
LIBRARY	Creates or modifies a macro library or a library of object modules.
LINK	Binds one or more object modules into an executable or shareable program image.
LINK/RSX11	Invokes the RSX-11M Task Builder to link one or more object modules into an executable task image.
MACRO	Invokes the VAX-11 MACRO assembler to assemble a set of MACRO language source statements.

COMMANDS FOR PROGRAM DEVELOPMENT AND CONTROL

MACRO/RSX11	Invokes the MACRO-11 assembler to assemble a set of assembler language source statements.
PATCH	Updata an image file.
RUN (Image)	Places an executable image in execution in the current process.
RUN (Process)	Creates a separate process to execute a specified image.
SHOW LOGICAL	Displays on the terminal the current assignments of logical names and equivalence names made by the ASSIGN, ALLOCATE, DEFINE or MOUNT commands.
SHOW PROCESS	Displays information about the current process, including subprocesses, privileges, quotas, and accounting information.
SHOW STATUS	Displays information about the image currently executing in the process.
SHOW SYSTEM	Displays the current status of all processes in the system.
SHOW TRANSLATION	Displays the result of logical name translation of a specific logical name.
STOP (Image)	Halts execution of a command procedure, program, or a subprocess.

COMMANDS FOR COMMAND PROCEDURES AND BATCH JOBS

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

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

UETP OPERATING INSTRUCTION SUMMARY

USER ENVIRONMENT TEST PACKAGE (UETP)

1. Log out from the field service account:

   $ LOGOUT

   The system responds:

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

2. Log in to the SYSTEST account:

   <CR>
   USERNAME: SYSTEST
   PASSWORD: UETP

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

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

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

   Note

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

For each magnetic tape drive perform the following steps:

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

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

Press the ONLINE switch. Check the baud rates and terminal characteristics (these should still be set according to specifications given in section 4.16, above).

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

```
$ @UETP/OUTPUT=TESTDATA

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

ENTER NUMBER OF LOAD TEST USERS [D]: n
ENTER NUMBER OF COMPLETE UETP TEST RUNS [D]: 1
ENTER SCRATCH MAGTAPE (E.G. MT0:) OR A<CR> device-name

<CR>
```

NOTE

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

Guidelines for Selecting

Number of Load Test Users

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

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

The I/O device tests

The native mode utility tests

The system load test

The compatibility mode tests

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

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

then
       STOP            ; Terminate execution of the UETP.

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

PRINTING THE ERROR LOG FILE

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

Procedure:

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

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

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

```
$ DIRECTORY ERRLOG.SYS
```

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

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

4. Invoke the SYE utility by typing the command:

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

PRINTING THE ERROR LOG FILE

Prompts

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

Command Parameters

input-file-spec

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

When the user explicitly specifies an input file, SYE defaults any omitted fields to the defaults for FORTRAN unit 1 that is, the highest version of SYS$DISK:[default-directory]FOR001.DAT.

output-file-spec

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

If the user explicitly specifies an output file, SYE defaults any omitted fields to the defaults for FORTRAN unit 2, that is, the highest version of SYS$DISK:[default-directory]FOR002.DAT.

report-option

One of the following report options:

R	Roll up
C	Cryptic
B	Brief
S	Standard
U	Unknown

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

The five report options are explained in the Description section.

device-name:

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

SYE prompts for the device name by typing

```
device name:      [<all>]        ?
```

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

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

SYE will accept generic device names, allowing the operator to specify that errors be reported for all devices of a particular type (e.g., DB:), for devices attached to a particular controller (e.g., DBA:), or for a particular device (e.g., DBA1:).

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

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

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

For example:

```
device name:     [<all>]    ? -DMA1:
```

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

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

date time

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

By default, SYE reports errors after 17-NOV-1853 and before 31-DEC-9999.

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

```
ERRLOG.OLD
```

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

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

```
$ PRINT filename
```

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

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

Example:

```
$ SET DEFAULT SYS$SYSTEM
$ SET DEFAULT SYS$DISK:[SYSERR]        $ SHOW DEFAULT
$ DIRECTORY ERRLOG.SYS                 $ DBB2:[SYSERR]

DIRECTORY DBB2:[SYSERR]
18-JUL-78 15:13

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

TOTAL OF 14./18. BLOCKS IN 1. FILE
$ RENAME ERRLOG.SYS ERRLOG.OLD/NEW_VERSION
$ RUN SYS$SYSTEM:SYE
SYE X0.6-0
_input file:   [SYSERR]ERRLOG.OLD   ?<CR>
_output file:  [SYS$OUTPUT]         ?ERRLOG.DAT
_options:      [ROLL-UP]            ?R
_device name:  [<all>]               ?<CR>
_after date:   [17-NOV-1858]        ?<CR>
                                     17-NOV-1858 00:00:00.00
_before date:  [31-DEC-9999]       ?<CR>
                                     31-DEC-9999 23:59:59.
```

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

The SET DEFAULT commands set the operator's default disk and directory to DBB2:[SYSERR]; the response to the SHOW DEFAULT command verifies this. The DIRECTORY command lists, on the operator's terminal, all the ERRLOG.SYS files contained in the [SYSERR] directory; ERRLOG.SYS;1 file is the only file there. The RENAME command renames ERRLOG.SYS to ERRLOG.OLD. The /NEW_VERSION qualifier requests that ERRLOG.OLD be assigned a new version number if a file of this name already exists. The operator then invokes the SYE utility by typing RUN SYS$SYSTEM:SYE. SYE prompts for six parameters. The first prompt requests the name of the file to be manipulated by SYE; the response is <CR>, which forces the default of ERRLOG.OLD. The second prompt requests the file to contain the error long report; the response is ERRLOG.DAT. The third prompt requests the type of report that SYE generates; the response is the ROLL UP (R) report. The fourth prompt requests the devices on which SYE reports errors; the response, <CR> indicates SYE should report errors on all devices. The fifth and sixth prompts request the time range in which errors are recorded; the responses <CR> indicate that SYE should report all errors that occur between 17-NOV-1858 and 31-DEC-9999. SYE then creates the error log report and notifies the operator at completion. The operator can then issue the PRINT command to obtain a hard copy of this report.

PRINTING THE ERROR LOG FILE

SYE Report types

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

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

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

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

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

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

Additional SYE Notes

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

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

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

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

R0=UBA Configuration Register

R1=UBA Control Register

R2=UBA Status Register

R3=UBA Diagnostic Control Register

R4=UBA Failed Map Address Register

R5=UBA Failed Unibus Address Register

5. TU45 Magtape errors may be reported as TE16 errors.

SECTION 8
CONVERSION TABLES
AND INTEGRATED CIRCUIT DIAGRAMS

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

HEX SUBTRACTER

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

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
20	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
30	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
40	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
50	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
60	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
70	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
80	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
90	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D0	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
10	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37
20	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57
30	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77
40	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117
50	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137
60	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157
70	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177
80	200	201	202	203	204	205	206	207	210	211	212	213	214	215	216	217
90	220	221	222	223	224	225	226	227	230	231	232	233	234	235	236	237
A0	240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257
B0	260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277
C0	300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317
D0	320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337
E0	340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357
F0	360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	377

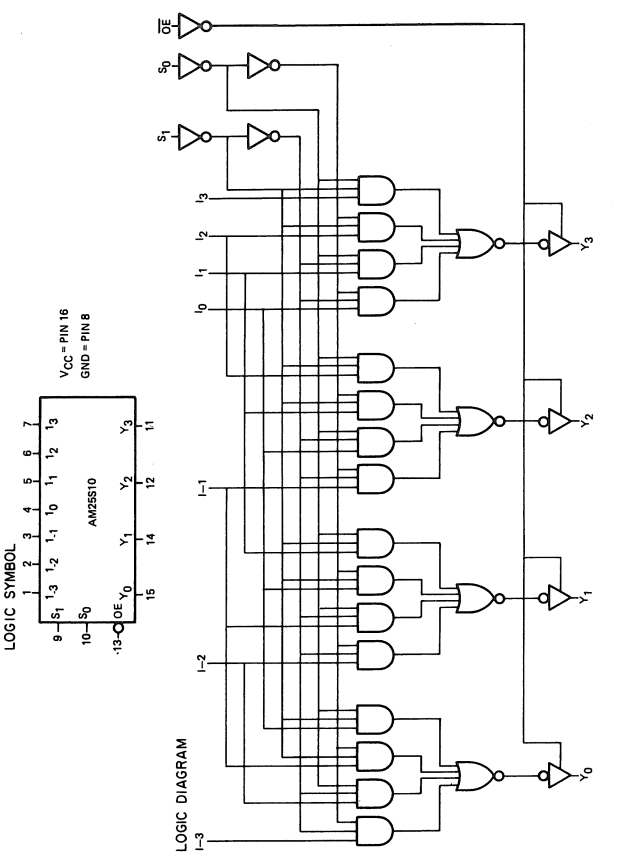
OCTAL/DECIMAL CONVERSION

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

HEXADECIMAL/ASCII CONVERSION

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

25S10 FOUR BIT SHIFTER CHIP WITH TRISTATE OUTPUT



TRUTH TABLE

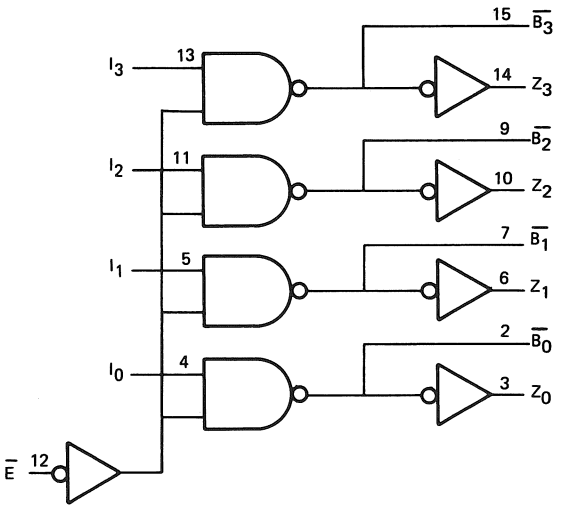
$\overline{OE}$	S_1	S_0	I_3	I_2	I_1	I_0	I_{-1}	I_{-2}	I_{-3}	Y_3	Y_2	Y_1	Y_0
H	X	X	X	X	X	X	X	X	X	Z	Z	Z	Z
L	L	L	D_3	D_2	D_1	D_0	X	X	X	D_3	D_2	D_1	D_0
L	L	H	X	D_2	D_1	D_0	D_{-1}	X	X	D_2	D_1	D_0	D_{-1}
L	H	L	X	X	D_1	D_0	D_{-1}	D_{-2}	X	D_1	D_0	D_{-1}	D_{-2}
L	H	H	X	X	X	D_0	D_{-1}	D_{-2}	D_{-3}	D_0	D_{-1}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

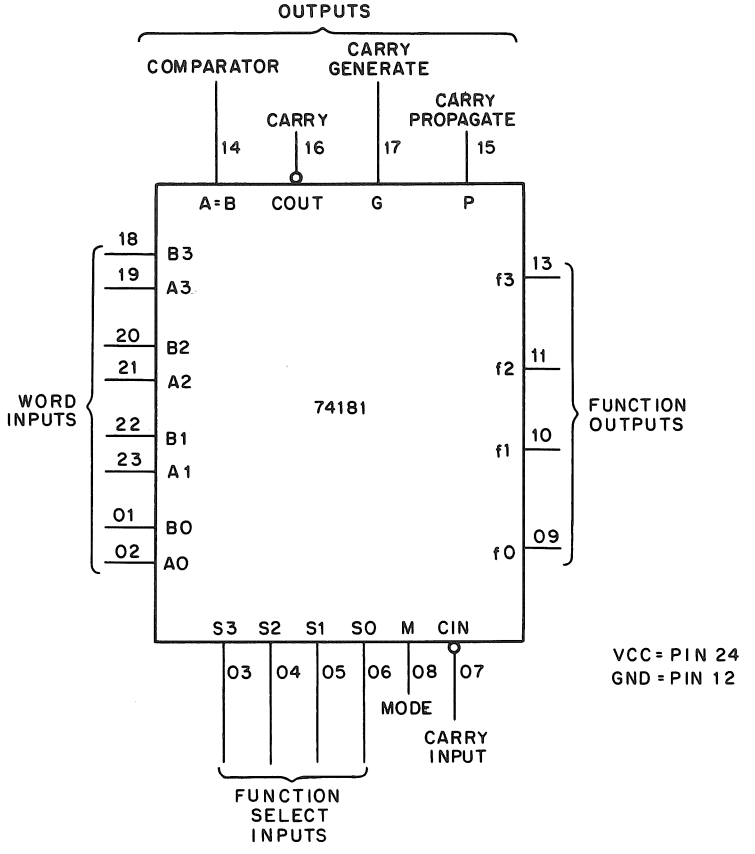


OUTPUTS		INPUTS	
$\overline{E}$	D	$\overline{B}$	Z
L	L	H	L
L	H	L	H
H	X	Y	$\overline{Y}$

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	$f = A \oplus \overline{B}$	$f = A \oplus B$
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	$f = A \oplus B$	$f = A \oplus \overline{B}$
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 $[A + \overline{B}]$	$f = A \text{ plus } A\overline{B}$
L	H	L	H	f = AB plus $[A + \overline{B}]$	$f = [A + B] \text{ plus } A\overline{B}$
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]	$f = [A + \overline{B}] \text{ plus } AB$
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	$f = [A + \overline{B}] \text{ plus A}$
H	H	H	L	$f = A\overline{B}$ plus A	$f = [A + \overline{B}] \text{ plus A}$
H	H	H	H	f = A	f = A minus 1

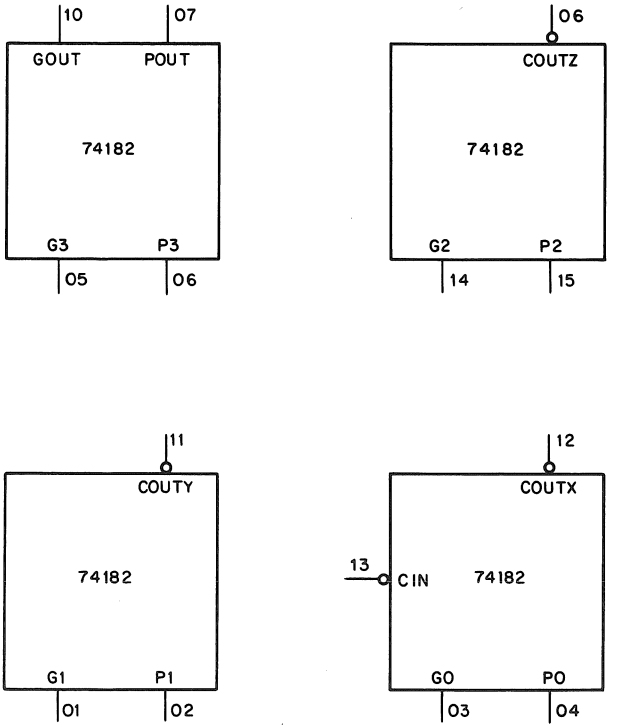
With mode control (M) and C_{in} low
† Each bit is shifted to the next more significant position.

IC-74181B

74182 LOOK AHEAD CARRY CHIP

PIN DESIGNATIONS

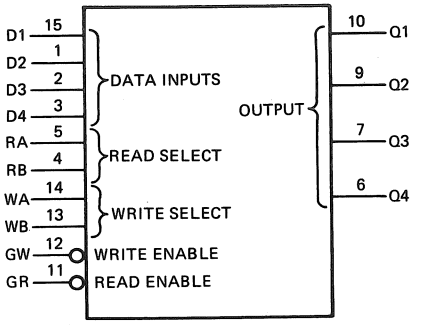
Designation	Pin No.	Function
G0, G1, G2, G3	3, 1, 14, 5	ACTIVE-LOW CARRY GENERATE INPUTS
P0, P1, P2, P3	4, 2, 15, 6	ACTIVE-LOW CARRY PROPAGATE INPUTS
CIN	13	CARRY INPUT
COUTX, COUTY, COUTZ	12, 11, 9	CARRY OUTPUTS
GOUT	10	ACTIVE-LOW CARRY GENERATE OUTPUT
POUT	7	ACTIVE-LOW CARRY PROPAGATE OUTPUT
V_{CC}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



VCC = PIN 16
GND = PIN 08

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

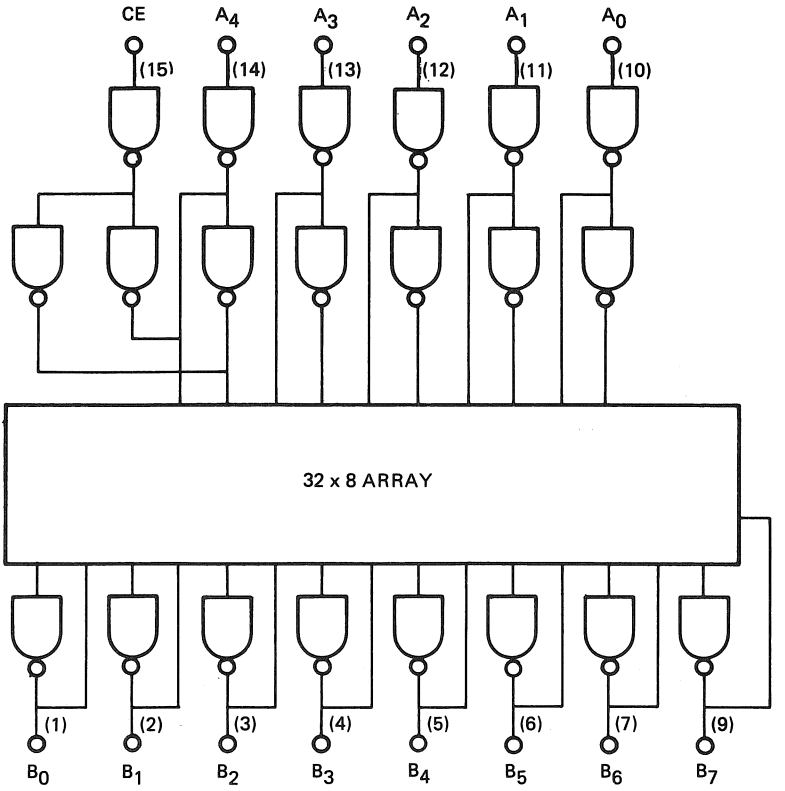
READ FUNCTION TABLE
(SEE NOTES A AND D)

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.

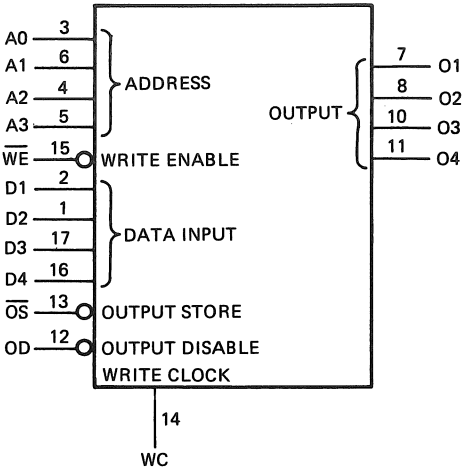
V_{CC} = (16)

GND = (8)

(N) = DENOTES PIN NUMBERS

IC-82S23
82S123

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



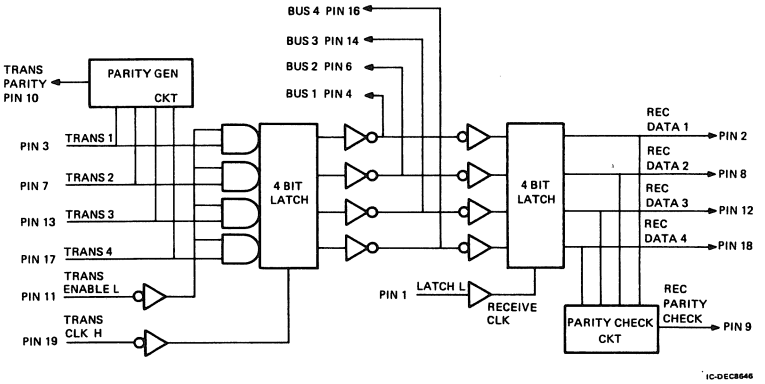
VCC 18
GND 9

TRUTH TABLE

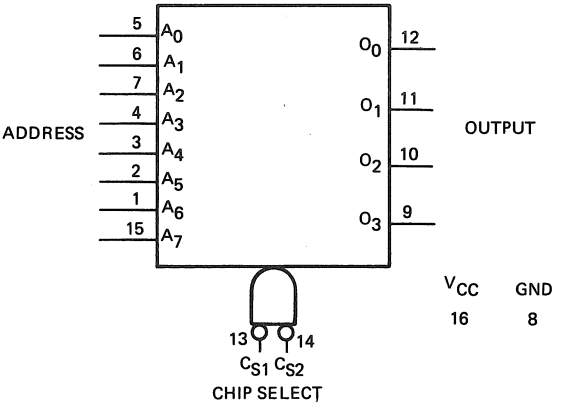
OD	$\overline{WE}$	CLK	$\overline{OS}$	MODE	OUTPUTS
L	X	X	L	OUTPUT STORE	DATA FROM LAST ADDRESSED LOCATION
X	L	⌐	X	WRITE DATA	DEPENDENT ON STATE OF OD AND $\overline{OS}$
L	X	X	H	READ DATA	DATA STORED IN ADDRESSED LOCATION
H	X	X	L	OUTPUT STORE	Hi-Z
H	X	X	H	OUTPUT DISABLE	Hi-Z

IC-85568

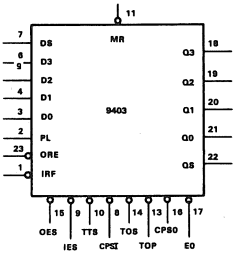
DEC 8646 4 BIT TRISTATE BACKPLANE INTERCONNECT TRANSCEIVER CHIP



93406 1024 BIT ROM CHIP

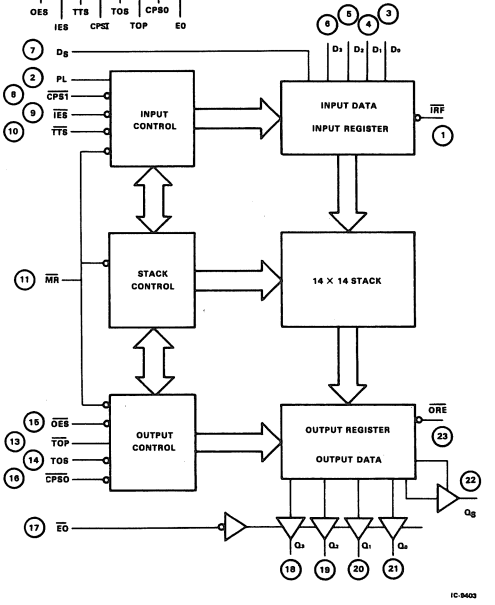


9403 FIFO BUFFER CHIP

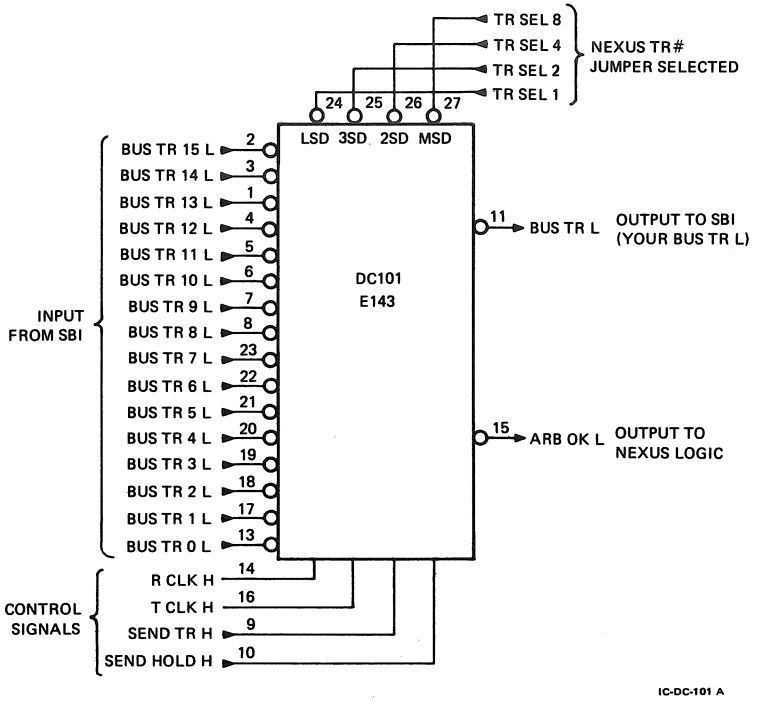


$D_0 - D_3$	Parallel Data Inputs
D_S	Serial Data Input
PL	Parallel Load Input
$\overline{CPSI}$	Serial Input Clock (Operates on Negative-Going Transition)
$\overline{IES}$	Serial Input Enable (Active LOW)
$\overline{TTS}$	Transfer to Stack Input (Active LOW)
$\overline{OES}$	Serial Output Enable Input (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)

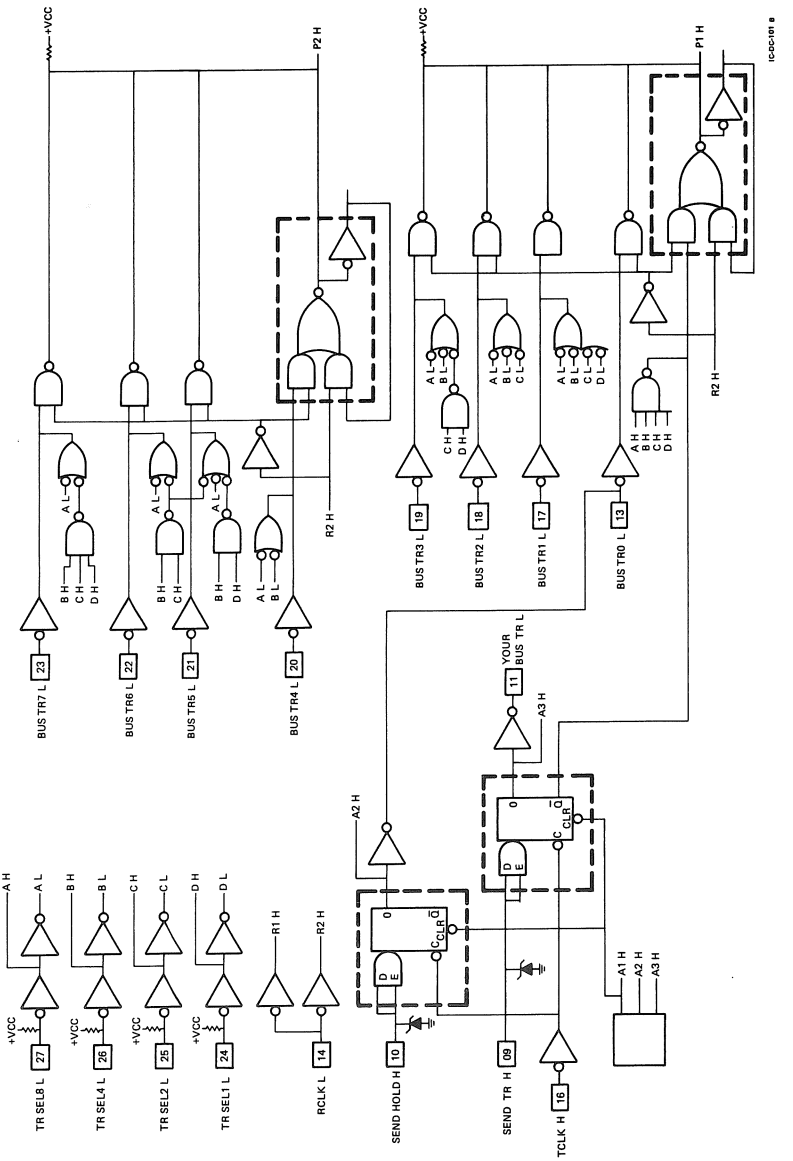
VDD = Pin 24
VSS = Pin 12
O = Pin Number



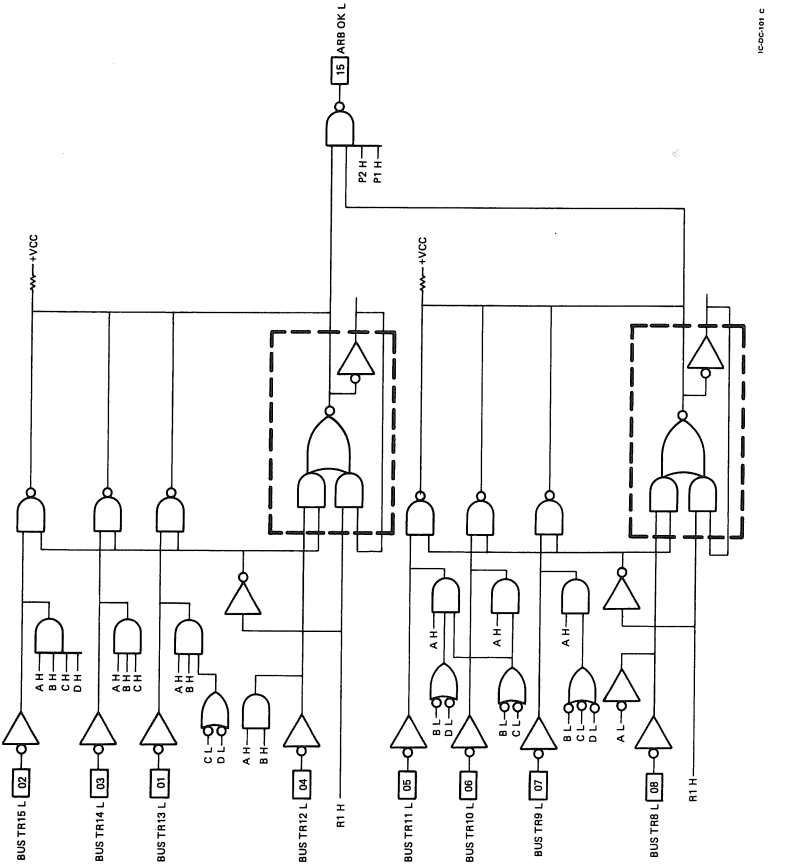
DC101 ARBITRATOR CHIP, PART 1



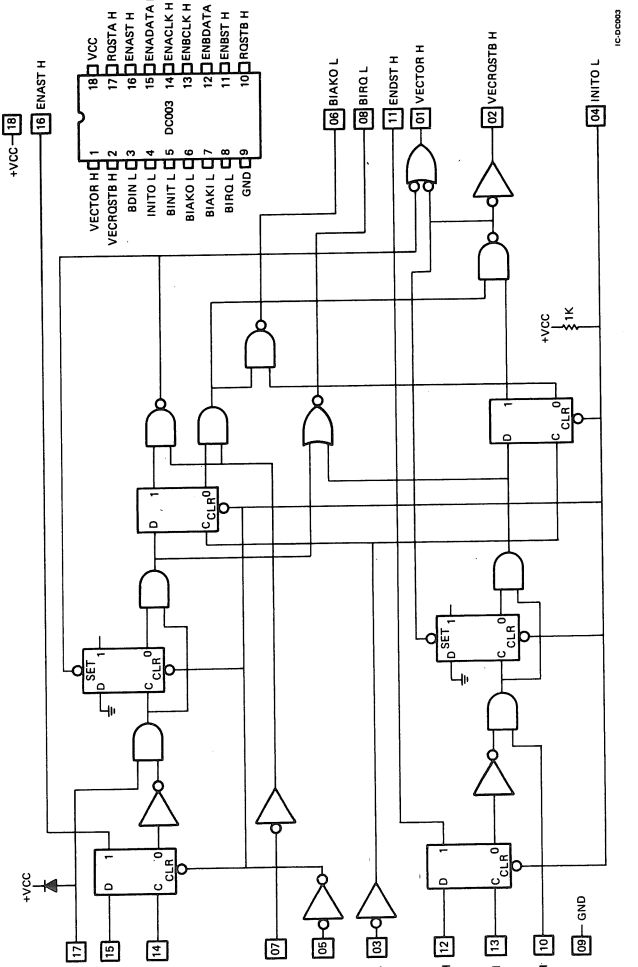
DC101 ARBITRATOR CHIP, PART 2

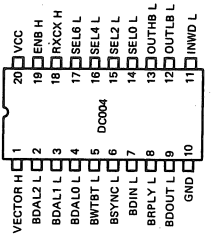


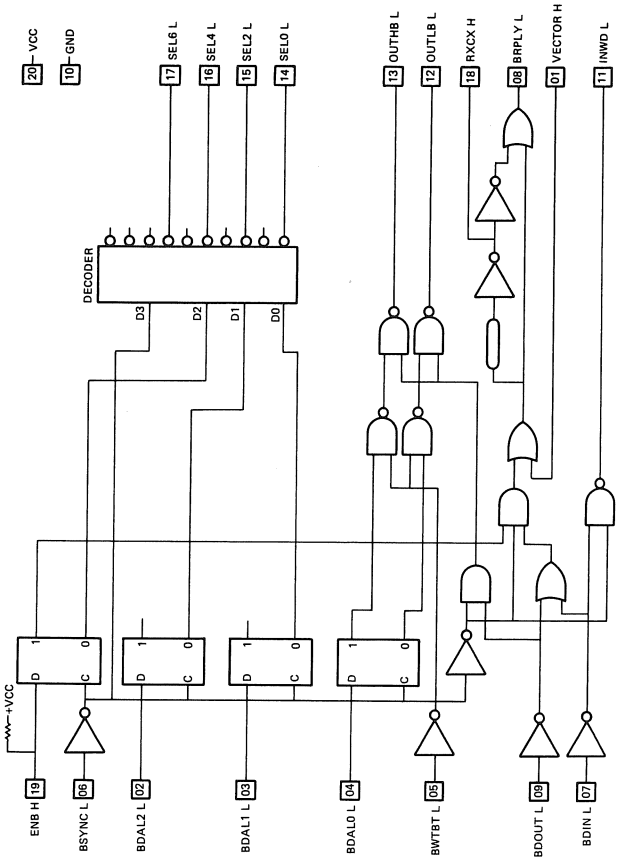
DC101 ARBITRATOR CHIP, PART 3



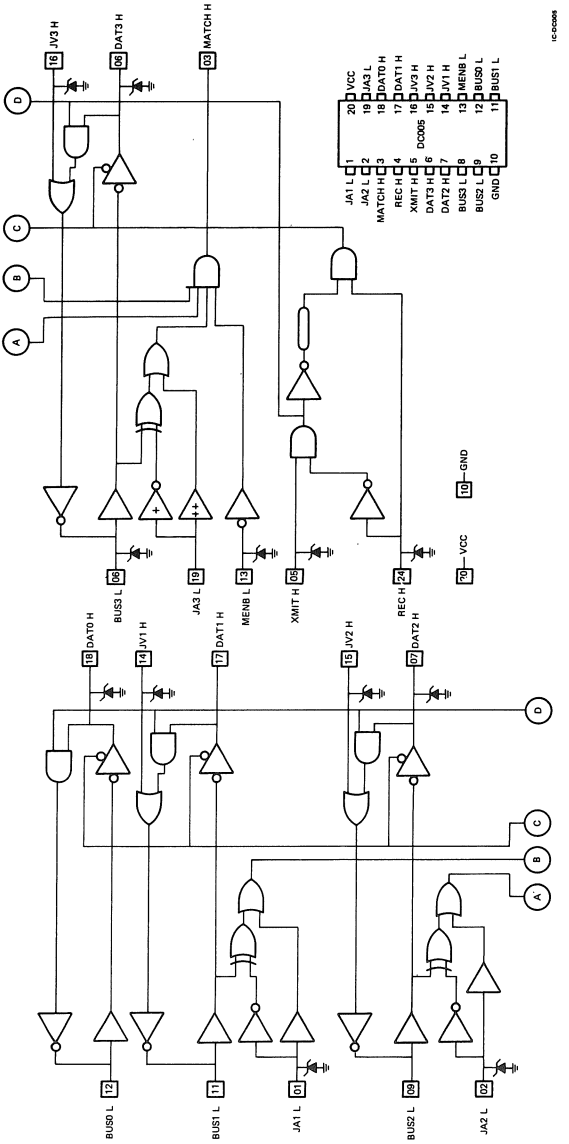
DC003 INTERRUPT CHIP




VECTOR H 1
BDAL2 L 2
BDAL1 L 3
BDAL0 L 4
BWTBT L 5
BSYNC L 6
BDIN L 7
BRPLY L 8
BDOUT L 9
GND 10
DC004
20 VCC
19 ENB H
18 RXCX H
17 SEL6 L
16 SEL4 L
15 SEL2 L
14 SEL0 L
13 OUTHB L
12 OUTLB L
11 INWD L
1C-DC004


20 VCC
10 GND
17 SEL6 L
16 SEL4 L
15 SEL2 L
14 SEL0 L
13 OUTHB L
12 OUTLB L
18 RXCX H
08 BRPLY L
01 VECTOR H
11 INWD L
DECODER
D3
D2
D1
D0
+VCC
19 ENB H
06 BSYNC L
02 BDAL2 L
03 BDAL1 L
04 BDAL0 L
05 BWTBT L
09 BDOUT L
07 BDIN L

DC005 TRANSCEIVER CHIP



NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

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

Reader's Comments

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

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

_____

_____

_____

What features are most useful? _____

_____

_____

_____

What faults or errors have you found in the manual? _____

_____

_____

_____

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

Does it satisfy *your* needs? _____ Why? _____

_____

_____

_____

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

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

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, Ma 01532
Attention: Communications Services (NR2/M15)
Customer Services Section

Order No. EK-11780-PG-001

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

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

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
Maynard, Massachusetts 01754

digital