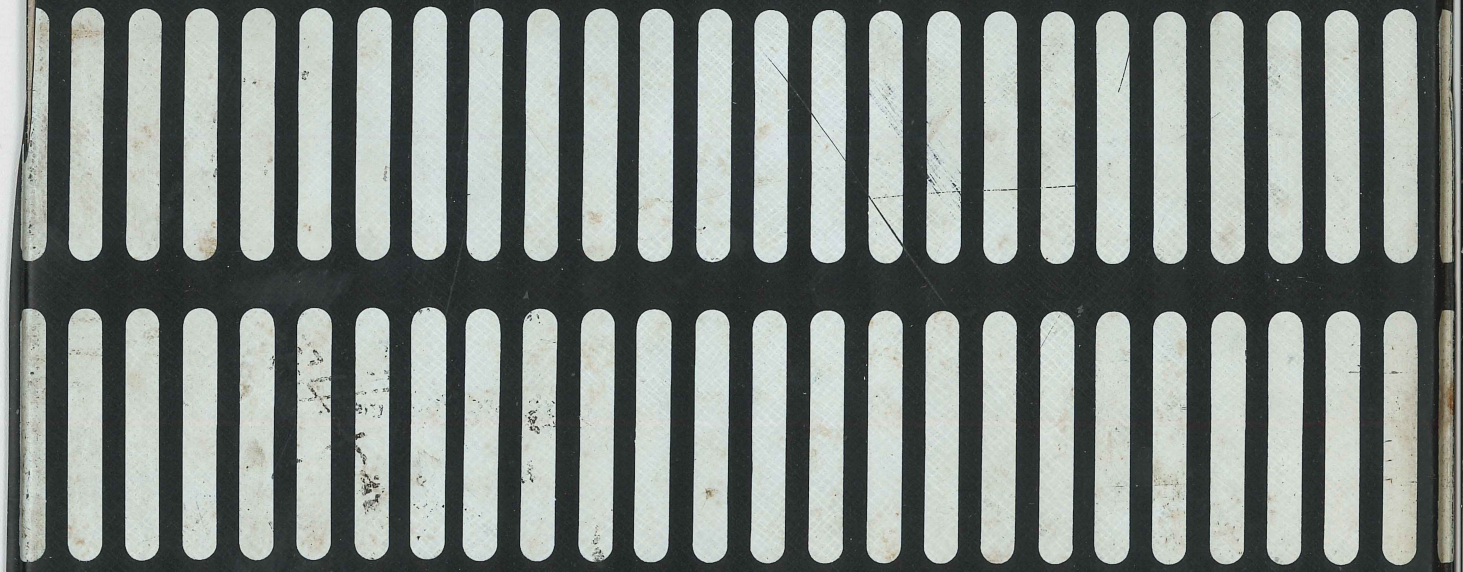
Alan Rose

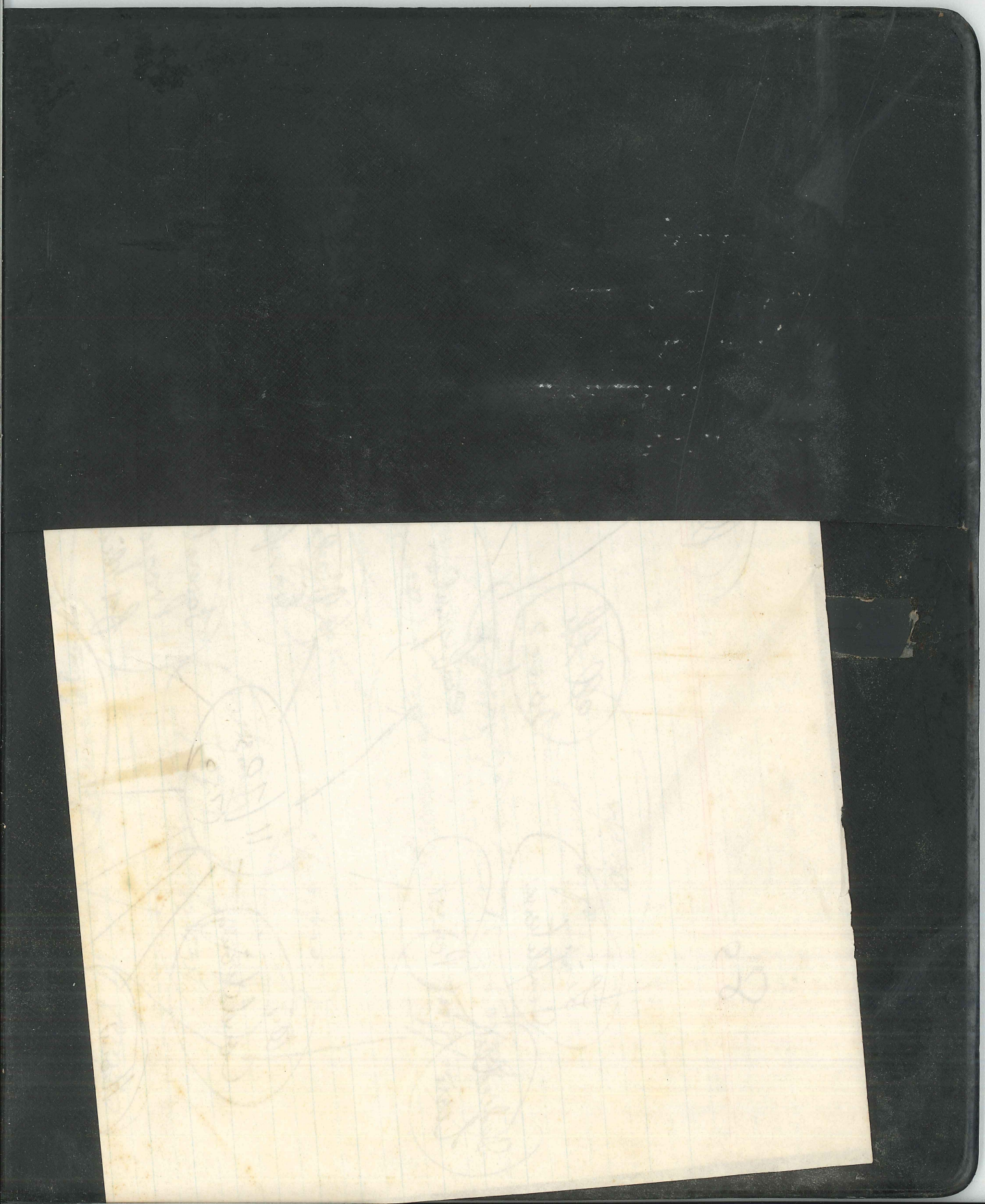
EK-VAXV3-HB-001

VAX Maintenance Handbook

VAX-11/750



digital



EK-VAXV1-HB-002

VAX Maintenance Handbook

VAX Systems

1983 Edition

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, July 1982
Second Edition, March 1983

Printed in U.S.A.

CONTENTS

CHAPTER 5 OPERATING SYSTEM

CHAPTER 6 DIAGNOSTICS

CHAPTER 7 I/O OPTIONS

CHAPTER 8 MISCELLANEOUS

CHAPTER 1
INTRODUCTION

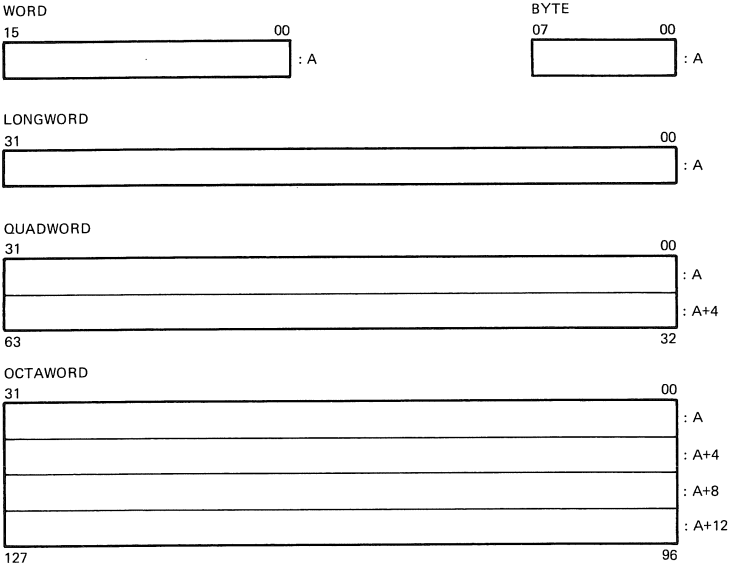
INTRODUCTION

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

This first volume of the VAX Maintenance Handbook provides a useful summary of system information that applies to all VAX systems. Each of the other volumes making up this series is processor specific; that is, it provides a highly compressed compilation of processor data that supplements hardware and software information available from such sources as manuals, print sets, microcode lists, and program tabulations.

CHAPTER 2
INSTRUCTION SET

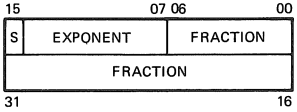
DATA TYPE REPRESENTATIONS



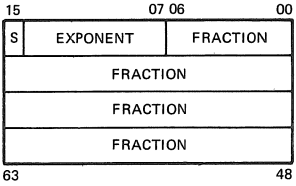
TK-8350

DATA TYPE REPRESENTATIONS (CONT)

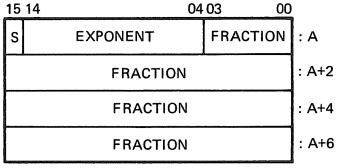
F_FLOATING



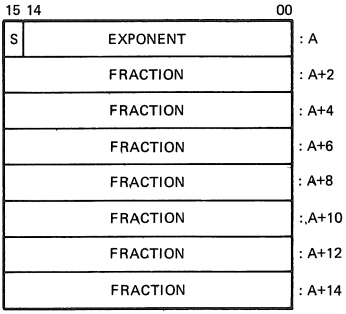
D_FLOATING



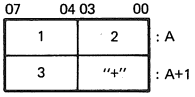
G_FLOATING



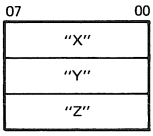
H_FLOATING



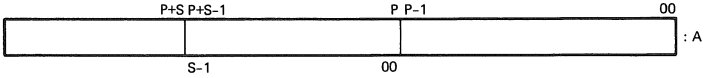
PACKED DECIMAL STRING (+123)



CHARACTER STRING (XYZ)



VARIABLE-LENGTH BIT FIELD



A=ADDRESS

TK-8351

GENERAL REGISTER ADDRESSING

7 6 5 4	3 2 1 0	Dec	Name	Assembler	r m w a v	PC	SP	Indexable?
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

VAX-11 SPECIAL REGISTER USAGE

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

SINGLE-BYTE OPCODE

LSB / MSB	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ø	HALT	NOP	REI	BPT	RET	RSB	LDPCTX	SVPCTX	CVTPS	CVTSP	INDEX	CRC	PROBER	PROBEW	INSQUE	REMQUE
1	BSBB	BRB	BNEQ	BEQL	BGTR	BLEQ	JSB	JMP	BGEQ	BLSS	BGTRU	BLEQU	BVC	BVS	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				INSQHI	INSQTI	REMQHI	REMQTI
6	ADDD2	ADDD3	SUBD2	SUBD3	MULD2	MULD3	DIVD2	DIVD3	CVTDB	CVTDW	CVTDL	CVTRDL	CVTBD	CVTWD	CVTLD	ACBD
7	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
D	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

TWO-BYTE OPCODES XXFD

LSB / MSB	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ø																
1																
2																
3			CVTDH	CVTGH												
4	ADDG2	ADDG3	SUBG2	SUBG3	MULG2	MULG3	DIVG2	DIVG3	CVTGB	CVTGW	CVTGL	CVTRGL	CVTBG	CVTWG	CVTLG	ACBG
5	MOVG	CMPG	MNEGG	TSTG	EMODG	POLYG	CVTGH									
6	ADDH2	ADDH3	SUBH2	SUBH3	MULH2	MULH3	DIVH2	DIVH3	CVTHB	CVTHW	CVTHL	CVTRHL	CVTBH	CVTWH	CVTLH	ACBH
7	MOVH	CMPH	MNEGH	TSTH	EMODH	POLYH	CVTHG						CLRH		MOVAH	PUSHAH
7													CLRO	MOVO	MOVAO	PUSHAO
8																
9									CVTFH	CVTFG						
A																
B																
C																
D																
E																
F							CVTHF	CVTHD								

VAX-11 INSTRUCTION SET BY OPCODE (CONT)

TWO-BYTE OPCODES XXFF

LSB / MSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0														BUGL	BUGW	
1																
2																
3																
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

VAX-11 INSTRUCTION SET 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 - Letter denoting the data type or the operand:

   B - Byte
   D - Double floating
   F - Floating
   G - G_floating
   H - H_floating
   L - Longword
   O - Octaword
   Q - Quadword
   W - Word

Instructions		Condition Codes			
		N	Z	V	C
9D	ACBB ADD COMPARE AND BRANCH BYTE LIMIT.RB, ADD.RB, INDEX.MB, DISPL.BW	INDEX<Ø	INDEX=Ø	INTEGER OVERFLOW	C
6F	ACBD ADD COMPARE AND BRANCH DOUBLE LIMIT.RD, ADD.RD, INDEX.MD, DISPL.BW	INDEX<Ø	INDEX=Ø	FLOATING OVERFLOW	C
4F	ACBF ADD COMPARE AND BRANCH FLOATING LIMIT.RF, ADD.RF, INDEX.MF, DISPL.BW	INDEX<Ø	INDEX=Ø	FLOATING OVERFLOW	C
4FFD	ACBG ADD COMPARE AND BRANCH G_FLOATING LIMIT.RG, ADD.RG, INDEX.MG, DISPL.BW	INDEX<Ø	INDEX=Ø	FLOATING OVERFLOW	C
6FFD	ACBH ADD COMPARE AND BRANCH H_FLOATING LIMIT.RH, ADD.RH, INDEX.RG, DISPL.BW	INDEX<Ø	INDEX=Ø	FLOATING OVERFLOW	C
F1	ACBL ADD COMPARE AND BRANCH LONG LIMIT.RL, ADD.RL, INDEX.ML, DISPL.BW	INDEX<Ø	INDEX=Ø	INTEGER OVERFLOW	C
3D	ACBW ADD COMPARE AND BRANCH WORD LIMIT.RW, ADD.RW, INDEX.MW, DISPL.BW	INDEX<Ø	INDEX=Ø	INTEGER OVERFLOW	C
58	ADAWI ADD ALIGNED WORD INTERLOCKED ADD.RL, SUM.ML	SUM<Ø	SUM=Ø	INTEGER OVERFLOW	C FROM MSB
8Ø	ADDB2 ADD BYTE 2 OPERAND ADD.RB, SUM.MB	SUM<Ø	SUM=Ø	INTEGER OVERFLOW	C FROM MSB
81	ADDB3 ADD BYTE 3 OPERAND ADD1.RB, ADD2.RB, SUM.WB	SUM<Ø	SUM=Ø	INTEGER OVERFLOW	C FROM MSB
6Ø	ADDD2* ADD DOUBLE 2 OPERAND ADD.RD, SUM.MD	SUM<Ø	SUM=Ø	FLOATING OVERFLOW	Ø
61	ADDD3* ADD DOUBLE 3 OPERAND ADD1.RD, ADD2.RD, SUM.WD	SUM<Ø	SUM=Ø	FLOATING OVERFLOW	Ø
4Ø	ADDF2 ADD FLOATING 2 OPERAND ADD.RF, SUM.MF	SUM<Ø	SUM=Ø	FLOATING OVERFLOW	Ø
41	ADDF3 ADD FLOATING 3 OPERAND ADD1.RF, ADD2.RF, SUM.WF	SUM<Ø	SUM=Ø	FLOATING OVERFLOW	Ø
4ØFD	ADDG2 ADD G_FLOATING 2 OPERAND ADD.RG, SUM.MG	SUM<Ø	SUM=Ø	FLOATING OVERFLOW	Ø

*ADDD is ADD D_FLOATING.

Instructions		Condition Codes			
		N	Z	V	C
41FD	ADDG3 ADD G_FLOATING 3 OPERAND ADD1.RG, ADD2.RG, SUM.WG	SUM<Ø	SUM=Ø	FLOATING OVERFLOW	Ø
60FD	ADDH2 ADD H_FLOATING 2 OPERAND ADD.RG, SUM.MG	SUM<Ø	SUM=Ø	FLOATING OVERFLOW	Ø
61FD	ADDH3 ADD H_FLOATING 3 OPERAND ADD1.RG, ADD2.RG, SUM.WG	SUM<Ø	SUM=Ø	FLOATING OVERFLOW	Ø
C0	ADDL2 ADD LONG 2 OPERAND ADD.RL, SUM.ML	SUM<Ø	SUM=Ø	INTEGER OVERFLOW	C FROM MSB
C1	ADDL3 ADD LONG 3 OPERAND ADD1.RL, ADD2.RL, SUM.WL	SUM<Ø	SUM=Ø	INTEGER OVERFLOW	C FROM MSB
20	ADDP4 ADD PACKED 4 OPERAND ADDLEN.RW, ADDADDR.AB, SUMLEN.RW, SUMADDR.AB	SUM STRING<Ø	SUM STRING=Ø	DECIMAL OVERFLOW	Ø
21	ADDP6 ADD PACKED 6 OPERAND ADD1LEN.RW, ADD1ADDR.AB, ADD2LEN.RW, ADD2ADDR.AB, SUMLEN.RW, SUMADDR.AB	SUM STRING<Ø	SUM STRING=Ø	DECIMAL OVERFLOW	Ø
A0	ADDW2 ADD WORD 2 OPERAND ADD.RW, SUM.MW	SUM<Ø	SUM=Ø	INTEGER OVERFLOW	C FROM MSB
A1	ADDW3 ADD WORD 3 OPERAND ADD.RW, ADD2.RW, SUM.WW	SUM<Ø	SUM=Ø	INTEGER OVERFLOW	C FROM MSB
D8	ADWC ADD WITH CARRY ADD.RL, SUM.ML	SUM<Ø	SUM=Ø	INTEGER OVERFLOW	C FROM MSB
F3	AOBLEQ ADD ONE AND BRANCH ON LESS OR EQUAL LIMIT.RL, INDEX.ML, DISPL.BB	INDEX<Ø	INDEX=Ø	INTEGER OVERFLOW	C
F2	AOBLSS ADD ONE AND BRANCH ON LESS LIMIT.RL, INDEX.ML, DISPL.BB	INDEX<Ø	INDEX=Ø	INTEGER OVERFLOW	C

Instructions		Condition Codes			
		N	Z	V	C
78	ASHL ARITHMETIC SHIFT LONG CNT.RB, SRC.RL, DST.WL	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
F8	ASHP ARITHMETIC SHIFT AND ROUNDED PACKED CNT.RB, SRCLEN.RW, SRCADDR.AB, ROUND.RB, DSTLEN.RB, DSTADDR.AB	DST STRING<Ø	DST STRING=Ø	DECIMAL OVERFLOW	Ø
79	ASHQ ARITHMETIC SHIFT QUAD CNT.RB, SRC.RQ, DST.WQ	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
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	BBSSI BRANCH ON BIT SET AND SET INTERLOCKED POS.RL, BASE.VB, DISPL.BB	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

Instructions		Condition Codes			
		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>Ø	DST=Ø	Ø	C
8B	BICB3 BIT CLEAR BYTE 3 OPERAND MASK.RB, SRC.RB, DST.WB	DST>Ø	DST=Ø	Ø	C
CA	BICL2 BIT CLEAR LONG 2 OPERAND MASK.RL, DST.ML	DST<Ø	DST=Ø	Ø	C
CB	BICL3 BIT CLEAR LONG 3 OPERAND MASK.RL, SRC.RL,DST.WL	DST<Ø	DST=Ø	Ø	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<Ø>
AA	BICW2 BIT CLEAR WORD 2 OPERAND MASK.RW, DST.MW	DST<Ø	DST=Ø	Ø	C
AB	BICW3 BIT CLEAR WORD 3 OPERAND MASK.RW, SRC.RW, DST.WW	DST<Ø	DST=Ø	Ø	C
88	BISB2 BIT SET BYTE 2 OPERAND MASK.RB, DST.MB	DST<Ø	DST=Ø	Ø	C
89	BISB3 BIT SET BYTE 3 OPERAND MASK.RB, SRC.RB, DST.WB	DST<Ø	DST=Ø	Ø	C

Instructions		Condition Codes			
		N	Z	V	C
C8	BISL2 BIT SET LONG 2 OPERAND MASK.RL, DST.ML	DST<Ø	DST=Ø	Ø	C
C9	BISL3 BIT SET LONG 3 OPERAND MASK.RL, SRC.RL, DST.WL	DST<Ø	DST=Ø	Ø	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<Ø>
A8	BISW2 BIT SET WORD 2 OPERAND MASK.RW, DST.MW	DST<Ø	DST=Ø	Ø	C
A9	BISW3 BIT SET WORD 3 OPERAND MASK.RW, SRC.RW, DST.WW	DST<Ø	DST=Ø	Ø	C
93	BITB BIT TEST BYTE MASK.RB, SRC.RB	TMP<Ø	TMP=Ø	Ø	C
D3	BITL BIT TEST LONG MASK.RL, SRC.RL	TMP<Ø	TMP=Ø	Ø	C
B3	BITW BIT TEST WORD MASK.RW, SRC.RW	TMP<Ø	TMP=Ø	Ø	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

Instructions		Condition Codes			
		N	Z	V	C
12	BNEQU BRANCH ON NOT EQUAL UNSIGNED DISPL.BB	N	Z	V	C
Ø3	BPT BREAK POINT TRAP	Ø	Ø	Ø	Ø
11	BRB BRANCH WITH BYTE DISPLACEMENT DISPL.BB	N	Z	V	C
31	BRW BRANCH WITH WORD DISPLACEMENT DISPL.BW	N	Z	V	C
1Ø	BSBB BRANCH TO SUBROUTINE WITH BYTE DISPLACEMENT DISPL.BB	N	Z	V	C
3Ø	BSBW BRANCH TO SUBROUTINE WITH WORD DISPLACEMENT DISPL.BW	N	Z	V	C
FDFF	BUGL BUGCHECK WITH LONGWORD MESSAGE IDENTIFIER MESSAGE.IDENTIFIER LONGWORD	N	Z	V	C
FEFF	BUGW BUGCHECK WITH WORD MESSAGE IDENTIFIER MESSAGE.IDENTIFIER.WORD	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	Ø	Ø	Ø	Ø
FB	CALLS CALL WITH STACK ARGUMENT LIST NUMARG.RL, DST.AB	Ø	Ø	Ø	Ø
8F	CASEB CASE BYTE SELECTOR.RB, BASE.RB, LIMIT.RL DISPL[Ø].BW,..., DISPL[LIMIT].BW	TEMP LSS LIMIT	TEMP EQL LIMIT	Ø	TEMP LSSU LIMIT

Instructions		Condition Codes			
		N	Z	V	C
CF	CASEL CASE LONG SELECTOR.RL, BASE.RL, LIMIT.RL, DISPL[Ø].BW,..., DISPL[LIMIT].BW	TEMP LSS LIMIT	TEMP EQL LIMIT	Ø	TEMP LSSU LIMIT
AF	CASEW CASE WORD SELECTOR.RW, BASE.RW, LIMIT.RW, DISPL[Ø].BW,..., DISPL[LIMIT].BW	TEMP LSS LIMIT	TEMP EQL LIMIT	Ø	TEMP LSSU LIMIT
BD	CHME CHANGE MODE TO EXECUTIVE CODE.RW	Ø	Ø	Ø	Ø
BC	CHMK CHANGE MODE TO KERNAL CODE.RW	Ø	Ø	Ø	Ø
BE	CHMS CHANGE MODE TO SUPERVISOR CODE.RW	Ø	Ø	Ø	Ø
BF	CHMU CHANGE MODE TO USER CODE.RW	Ø	Ø	Ø	Ø
94	CLRB CLEAR BYTE DST.WB	Ø	1	Ø	C
7C	CLRD CLEAR D_FLOATING DST.WQ	Ø	1	Ø	C
D4	CLRF CLEAR F_FLOATING DST.WL	Ø	1	Ø	C
7C	CLRG CLEAR G_FLOATING DST.WX	Ø	1	Ø	C
7CFD	CLRH CLEAR H_FLOATING DST.WH	Ø	1	Ø	C
D4	CLRL CLEAR LONG DST.WL	Ø	1	Ø	C
7CFD	CLRO CLEAR OCTAWORD DST.WO	Ø	1	Ø	C
7C	CLRQ CLEAR QUAD DST.WQ	Ø	1	Ø	C
B4	CLRW CLEAR WORD DST.WW	Ø	1	Ø	C

Instructions	Condition Codes			
	N	Z	V	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 D_FLOATING SRC1.RD, SRC2.RD	SRC1<SRC2	SRC1=SRC2	0	0
51 CMPF COMPARE F_FLOATING SRC1.RF, SRC2.RF	SRC1<SRC2	SRC1=SRC2	0	0
51FD CMPG COMPARE G_FLOATING SRC1.RG, SRC2.RG	SRC1<SRC2	SRC1=SRC2	0	0
71FD CMPH COMPARE H_FLOATING SRC1.RH, SRC2.RH	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 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

Instructions		Condition Codes			
		N	Z	V	C
ED	CMPZV COMPARE ZERO-EXTENDED FIELD POS.RL, SIZE.RB, BASE.VB, SRC.RL	TMP<SRC	TMP=SRC	Ø	TMP LSSU SRC
ØB	CRC CALCULATE CYCLIC REDUNDANCY CHECK TBL.AB, INICRC.RL, STRLEN.RW, STREAM.AB, RESULT=W	DST<Ø	DST=Ø	Ø	C
6C	CVTBD CONVERT BYTE TO D_FLOATING SRC.RB, DST.WD	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
4C	CVTBF CONVERT BYTE TO F_FLOATING SRC.RB, DST.WF	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
4CFD	CVTBG CONVERT BYTE TO G_FLOATING SRC.RB, DST.WG	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
6CFD	CVTBH CONVERT BYTE TO H_FLOATING SRC.RB, DST.WH	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
98	CVTBL CONVERT BYTE TO LONG SRC.RB, DST.WL	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
99	CVTBW CONVERT BYTE TO WORD SRC.RB, DST.WW	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
68	CVTDB CONVERT D_FLOATING TO BYTE SRC.RD, DST.WB	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
76	CVTDF CONVERT D_FLOATING TO F_FLOATING SRC.RD, DST.WF	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
32FD	CVTDH CONVERT D_FLOATING TO H_FLOATING SRC.RD, DST.WH	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø

Instructions		Condition Codes			
		N	Z	V	C
6A	CVTDL CONVERT D_FLOATING TO LONG SRC.RD, DST.WL	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
69	CVTDW CONVERT D_FLOATING TO WORD SRC.RD, DST.WW	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
48	CVTFB CONVERT D_FLOATING TO BYTE SRC.RF, DST.WB	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
56	CVTFD CONVERT F_FLOATING TO D_FLOATING SRC.RF, DST.WD	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
99FD	CVTFG CONVERT F_FLOATING TO G_FLOATING SRC.RF, DST.WG	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
98FD	CVTFH CONVERT F_FLOATING TO H_FLOATING SRC.RF, DST.WH	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
4A	CVTFL CONVERT F_FLOATING TO LONG SRC.RF, DST.WL	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
49	CVTFW CONVERT F_FLOATING TO WORD SRC.RF, DST.WW	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
48FD	CVTGB CONVERT G_FLOATING TO BYTE SRC.RG, DST.WB	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
33FD	CVTGF CONVERT G_FLOATING TO F_FLOATING SRC.RG, DST.WF	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø

Instructions		Condition Codes			
		N	Z	V	C
56FD	CVTGH CONVERT G_FLOATING TO H_FLOATING SRC.RG, DST.WH	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
4AFD	CVTGL CONVERT G_FLOATING TO LONGWORD SRC.RG, DST.WL	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
49FD	CVTGW CONVERT G_FLOATING TO WORD SRC.RG, DST.WW	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
68FD	CVTHB CONVERT H_FLOATING TO BYTE SRC.RH, DST.WB	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
F7FD	CVTHD CONVERT H_FLOATING TO D_FLOATING SRC.RH, DST.WD	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
F6FD	CVTHF CONVERT H_FLOATING TO F_FLOATING SRC.RH, DST.WF	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
76FD	CVTHG CONVERT H_FLOATING TO G_FLOATING SRC.RH, DST.WG	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
6AFD	CVTHL CONVERT H_FLOATING TO LONGWORD SRC.RH, DST.WL	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
69FD	CVTHW CONVERT H_FLOATING TO WORD SRC.RH, DST.WW	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
F6	CVTLB CONVERT LONG TO BYTE SRC.RL, DST.WB	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
6E	CVTLD CONVERT LONG TO D_FLOATING SRC.RL, DST.WD	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø

Instructions	Condition Codes			
	N	Z	V	C
4E CVTLF CONVERT LONG TO F_FLOATING SRC.RL, DST.WF	DST<Ø	DST=1	SRC CANNOT BE REPRESENTED IN DST	Ø
4EFD CVTLG CONVERT LONGWORD TO G_FLOATING SRC.RL, DST.WG	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
6EFD CVTLH CONVERT LONGWORD TO H_FLOATING SRC.RL, DST.WH	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
F9 CVTLP CONVERT LONG TO PACKED SRC.RL, DSTLEN.RW, DSTADDR.AB	DST STRING<Ø	DST STRING=Ø	DECIMAL OVERFLOW	Ø
F7 CVTLW CONVERT LONG TO WORD SRC.RL, DST.WW	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
36 CVTPL CONVERT PACKED TO LONG SRCLEN.RW, SRCADDR.AB, DST.WL	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
Ø8 CVTPS CONVERT PACKED TO LEADING SEPARATE NUMERIC SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB	SRC STRING<Ø	SRC STRING=Ø	DECIMAL OVERFLOW	Ø
24 CVTPT CONVERT PACKED TO TRAILING NUMERIC SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
6B CVTRDL CONVERT ROUNDED D_FLOATING TO LONG SRC.RD, DST.WL	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø

Instructions		Condition Codes			
		N	Z	V	C
4B	CVTRFL CONVERT ROUNDED F_FLOATING TO LONG SRC.RF, DST.WL	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
4BFD	CVTRGL CONVERT ROUNDED G_FLOATING TO LONGWORD SRC.RG, DST.WL	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
6BFD	CVTRHL CONVERT ROUNDED H_FLOATING TO LONGWORD SRC.RH, DST.WL	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
Ø9	CVTSP CONVERT LEADING SEPARATE NUMERIC TO PACKED SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB	DST STRING<Ø	DST STRING=Ø	DECIMAL OVERFLOW	Ø
26	CVTTP CONVERT TRAILING NUMERIC TO PACKED SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB	DST<Ø	DST=Ø	DECIMAL OVERFLOW	Ø
33	CVTWB CONVERT WORD TO BYTE SRC.RW, DST.WB	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
6D	CVTWD CONVERT WORD TO D_FLOATING SRC.RW, DST.WD	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
4D	CVTWF CONVERT WORD TO F_FLOATING SRC.RW, DST.WF	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
4DFD	CVTWG CONVERT WORD TO G_FLOATING SRC.RW, DST.WG	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø
6DFD	CVTWH CONVERT WORD TO H_FLOATING SRC.RW, DST.WH	DST<Ø	DST=Ø	SRC CANNOT BE REPRESENTED IN DST	Ø

Instructions	Condition Codes			
	N	Z	V	C
32 CVTWL CONVERT WORD TO LONG SRC.RW, DST.WL	DST<Ø	DST=Ø	INTEGER OVERFLOW	Ø
97 DECB DECREMENT BYTE DIF.MB	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW INTO MSB
D7 DECL DECREMENT LONG DIF.ML	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW INTO MSB
B7 DECW DECREMENT WORD DIF.MW	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW INTO MSB
86 DIVB2 DIVIDE BYTE 2 OPERAND DIVR.RB, QUO.MB	QUO<Ø	QUO=Ø	INTEGER OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
87 DIVB3 DIVIDE BYTE 3 OPERAND DIVR.RB, DIVD.RB, QUO.WB	QUO<Ø	QUO=Ø	INTEGER OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
66 DIVD2 DIVIDE D_FLOATING 2 OPERAND DIVR.RD, QUO.MD	QUO<Ø	QUO=Ø	FLOATING OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
67 DIVD3 DIVIDE D_FLOATING 3 OPERAND DIVR.RD, DIVD.RD, QUO.WD	QUO<Ø	QUO=Ø	FLOATING OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
46 DIVF2 DIVIDE F_FLOATING 2 OPERAND DIVD.RF, QUO.MF	QUO<Ø	QUO=Ø	FLOATING OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
47 DIVF3 DIVIDE F_FLOATING 3 OPERAND DIVR.RF, DIVD.RF, QUO.WF	QUO<Ø	QUO=Ø	FLOATING OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
46FD DIVG2 DIVIDE G_FLOATING 2 OPERAND DIVR.RG, QUO.MG	QUO<Ø	QUO=Ø	FLOATING OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
47FD DIVG3 DIVIDE G_FLOATING 3 OPERAND DIVR.RG, DIVD.RG, QUO.WG	QUO<Ø	QUO=Ø	FLOATING OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø

Instructions		Condition Codes			
		N	Z	V	C
66FD	DIVH2 DIVIDE H_FLOATING 2 OPERAND DIVR.RH, QUO.MH	QUO<Ø	QUO=Ø	FLOATING OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
67FD	DIVH3 DIVIDE H_FLOATING 3 OPERAND DIVR.RH, DIVD.RH, QUO.WH	QUO<Ø	QUO=Ø	FLOATING OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
C6	DIVL2 DIVIDE LONG 2 OPERAND DIVR.RL, QUO.ML	QUO<Ø	QUO=Ø	INTEGER OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
C7	DIVL3 DIVIDE LONG 3 OPERAND DIVR.RL, DIVD.RL, QUO.WL	QUO<Ø	QUO=Ø	INTEGER OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
27	DIVP DIVIDE PACKED DIVRLEN.RW, DIVRADDR.AB, DIVDLEN.RW, DEVDADDR.AB, QUOLEN.RW, QUOADDR.AB	QUO STRING<Ø	QUO STRING=Ø	DECIMAL OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
A6	DIVW2 DIVIDE WORD 2 OPERAND DIVR.RW, QUO.MW	QUO<Ø	QUO=Ø	INTEGER OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
A7	DIVW3 DIVIDE WORD 3 OPERAND DIVR.RW, DIVD.RW, QUO.WW	QUO<Ø	QUO=Ø	INTEGER OVERFLOW OR (V <-- 1 IF DIVR=Ø)	Ø
38	EDITPC EDIT PACKED TO CHARACTER STRING SRCLEN.RW, SRCADDR.AB, PATTERN.AB, DSTADDR.AB	SRC STRING<Ø	SRC STRING=Ø	DECIMAL OVERFLOW	SIGNIFICANCE
7B	EDIV EXTENDED DIVIDE DIVR.RL, DIVD.RQ, QUO.WL, REM.WL	QUO<Ø	QUO=Ø	INTEGER OVERFLOW (V <-- 1 IF DIVR=Ø)	Ø
74	EMODD EXTENDED MODULUS D_FLOATING MULR.RD, MULRX.RB, MULD.RD, INT.WL, FRACT.WD	FRACT<Ø	FRACT=Ø	INTEGER OVERFLOW	Ø
54	EMODF EXTENDED MODULUS F_FLOATING MULR.RF, MULRX.RB, MULD.RF, INT.WL, FRACT.WF	FRACT<Ø	FRACT=Ø	INTEGER OVERFLOW	Ø

Instructions		Condition Codes			
		N	Z	V	C
54FD	EMODG EXTENDED MODULUS G_FLOATING MULR.RG, MULRG.RW, MULD.RG, INT.WL, FRACT.WG	FRACT<0	FRACT=0	INTEGER OVERFLOW	0
74FD	EMODH EXTENDED MODULUS H_FLOATING MULR.RH, MULRH.RW, MULD.RH, INT.WL, FRACT.WH	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
FD	ESCD RESERVED TO DIGITAL				
FE	ESCE RESERVED TO DIGITAL				
FF	ESCF RESERVED TO DIGITAL				
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

Instructions		Condition Codes			
		N	Z	V	C
ØA	INDEX INDEX COMPUTE SUBSCRIPT.RL, LOW.RL, HIGH.RL, SIZE.RL, INDEXIN.RL, INDEXOUT.WL	INDEXOUT<Ø	INDEXOUT=Ø	Ø	Ø
5C	INSQHI INSERT INTO QUEUE HEAD, INTERLOCKED OPCODE ENTRY.AB, HEADER.AQ	Ø	ENTRY=(ENTRY+4)	Ø	Ø
5D	INSQTI INSERT INTO QUEUE TAIL, INTERLOCKED OPCODE ENTRY.AB, HEADER.AQ	Ø	ENTRY=(ENTRY+4)	Ø	Ø
FØ	INSV INSERT FIELD SRC.RL, POS.RL, SIZE.RB, BASE.VB	Ø	Ø	Ø	C
ØE	INSQUE INSERT INTO QUEUE ENTRY.AB, PRED.AB	ENTRY<ENTRY+4	ENTRY=ENTRY+4	Ø	ENTRY LSSU ENTRY+4
17	JMP JUMP DST.AB	N	Z	V	C
16	JSB JUMP TO SUBROUTINE DST.AB	N	Z	V	C
Ø6	LDPCTX LOAD PROCESS CONTEXT	N	Z	V	C
3A	LOCC LOCATE CHARACTER CHAR.RB, LEN.RW, ADDR.AB	Ø	RØ=Ø	Ø	Ø
39	MATCHC MATCH CHARACTERS OBJLEN1.RW, OBJADDR1.AB, SRCLEN2.RW, SRCADDR2.AB	Ø	RØ=Ø	Ø	Ø
92	MCOMB MOVE COMPLEMENTED BYTE SRC.RB, DST.WB	DST<Ø	DST=Ø	Ø	C
D2	MCOML MOVE COMPLEMENTED LONG SRC.RL, DST.WL	DST<Ø	DST=Ø	Ø	C
B2	MCOMW MOVE COMPLEMENTED WORD SRC.RW, DST.WW	DST<Ø	DST=Ø	Ø	C

Instructions		Condition Codes			
		N	Z	V	C
DB	MFPR MOVE FROM PROCESSOR REGISTER SRC.RL, DST.WL	DST<Ø N	DST=Ø Z	Ø V	C (DST REG REPLACED) C (DST REG NOT REPLACED)
8E	MNEGB MOVE NEGATED BYTE SRC.RB, DST.WB	DST<Ø	DST=Ø	INTEGER OVERFLOW	DST≠Ø
72	MNEGD MOVE NEGATED D_FLOATING SRC.RD, DST.WD	DST<Ø	DST=Ø	Ø	Ø
52	MNEGF MOVE NEGATED FLOATING SRC.RF, DST.WF	DST<Ø	DST=Ø	Ø	Ø
52FD	MNEGG MOVE NEGATED G_FLOATING SRC.RG, DST.WG	DST<Ø	DST=Ø	Ø	C
72FD	MNEGH MOVE NEGATED H_FLOATING SRC.RH, DST.WH	DST<Ø	DST=Ø	Ø	Ø
CE	MNEGL MOVE NEGATED LONG SCR.RL, DST.WL	DST<Ø	DST=Ø	INTEGER OVERFLOW	DST≠Ø
AE	MNEGW MOVE NEGATED WORD SRC.RW, DST.WW	DST<Ø	DST=Ø	INTEGER OVERFLOW	DST≠Ø
9E	MOVAB MOVE ADDRESS OF BYTE SRC.AB, DST.WL	DST<Ø	DST=Ø	Ø	C
7E	MOVAD MOVE ADDRESS OF D_FLOATING SRC.AQ, DST.WL	DST<Ø	DST=Ø	Ø	C
DE	MOVAF MOVE ADDRESS OF F_FLOATING SRC.AL, DST.WL	DST<Ø	DST=Ø	Ø	C
7E	MOVAG MOVE ADDRESS OF G_FLOATING	DST<Ø	DST=Ø	Ø	C
7EFD	MOVAH MOVE ADDRESS OF H_FLOATING SRC.AH, DST.WL	DST<Ø	DST=Ø	Ø	C
DE	MOVAL MOVE ADDRESS OF LONG SRC.AL, DST.WL	DST<Ø	DST=Ø	Ø	C
7EFD	MOVAO MOVE ADDRESS OCTAWORD SRC.AO, DST.WL	DST<Ø	DST=Ø	Ø	C

Instructions		Condition Codes			
		N	Z	V	C
7E	MOVAQ MOVE ADDRESS OF QUAD SRC.AQ, DST.WL	DST<Ø	DST=Ø	Ø	C
3E	MOVAW MOVE ADDRESS OF WORD SRC.AW, DST.WL	DST<Ø	DST=Ø	Ø	C
9Ø	MOVB MOVE BYTE SRC.RB, DST.WB	DST<Ø	DST=Ø	Ø	C
28	MOVC3 MOVE CHARACTER 3 OPERAND LEN.RW, SRCADDR.AB, DSTADDR.AB SRCLEN=SOURCE LENGTH (DSTLEN=DESTINATION LENGTH)	Ø	1	Ø	Ø
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	Ø	SRCLEN LSSU DSTLEN
7Ø	MOVD MOVE D_FLOATING SRC.RD, DST.WD SRCLEN=SOURCE LENGTH (DSTLEN=DESTINATION LENGTH)	DST<Ø	DST=Ø	Ø	C
5Ø	MOVF MOVE F_FLOATING SRC.RF, DST.WF	DST<Ø	DST=Ø	Ø	C
5ØFD	MOVG MOVE G_FLOATING SRC.RG, DST.WG	DST<Ø	DST=Ø	Ø	C
7ØFD	MOVH MOVE H_FLOATING SRC.RH, DST.WH	DST<Ø	DST=Ø	Ø	C
DØ	MOVL MOVE LONG SRC.RL, DST.WL	DST<Ø	DST=Ø	Ø	C
7DFD	MOVO MOVE OCTAWORD SRC.RO, DST.WO	DST<Ø	DST=Ø	Ø	C
34	MOVP MOVE PACKED LEN.RW, SRCADDR.AB, DSTADDR.AB	DST<Ø	DST=Ø	Ø	C

Instructions		Condition Codes			
		N	Z	V	C
DC	MOVPSL MOVE PROGRAM STATUS LONGWORD DST.WL	N	Z	V	C
7D	MOVQ MOVE QUAD SRC.RQ, DST.WQ	DST STRING<Ø	DST STRING=Ø	Ø	C
2E	MOVTC MOVE TRANSLATED CHARACTERS SRCLEN.RW, SRCADDR.AB, FILL.RB, TBLADDR.AB, DSTLEN.RW, DSTADDR.AB	SRCLEN<DSTLEN	SRCLEN=DSTLEN	Ø	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
BØ	MOVW MOVE WORD SRC.RW, DST.WW	DST<Ø	DST=Ø	Ø	C
9A	MOVZBL MOVE ZERO-EXTENDED BYTE TO LONG SRC.RB, DST.WL	Ø	DST=Ø	Ø	C
9B	MOVZBW MOVE ZERO-EXTENDED BYTE TO WORD SRC.RB, DST.WW	Ø	DST=Ø	Ø	C
3C	MOVZWL MOVE ZERO-EXTENDED WORD TO LONG SRC.RW, DST.WL	Ø	DST=Ø	Ø	C
DA	MTPR MOVE TO PROCESSOR REGISTER SRC.RL, DST.RL	SRC<Ø N	SRC=Ø Z	Ø V	C (IF REG IS REPLACED) C (IF REG NOT REPLACED)
84	MULB2 MULTIPLY BYTE 2 OPERAND MULR.RB, PROD.WD	PROD<Ø	PROD=Ø	INTEGER OVERFLOW	Ø
85	MULB3 MULTIPLY BYTE 3 OPERAND MULR.RB, MULD.RB, PROD.WB	PROD<Ø	PROD=Ø	INTEGER OVERFLOW	Ø
64	MULD2 MULTIPLY D_FLOATING 2 OPERAND MULR.RD, PROD.MD	PROD<Ø	PROD=Ø	FLOATING OVERFLOW	Ø

Instructions		Condition Codes			
		N	Z	V	C
65	MULD3 MULTIPLY D_FLOATING 3 OPERAND MULR.RD, MULR.$\overline{RD}$, PROD.WD	PROD<Ø	PROD=Ø	FLOATING OVERFLOW	Ø
44	MULF2 MULTIPLY F_FLOATING 2 OPERAND MULR.RF, PROD.$\overline{MF}$	PROD<Ø	PROD=Ø	FLOATING OVERFLOW	Ø
45	MULF3 MULTIPLY F_FLOATING 3 OPERAND MULR.RF, MULD.$\overline{RF}$, PROD.WF	PROD<Ø	PROD=Ø	FLOATING OVERFLOW	Ø
44FD	MULG2 MULTIPLY G_FLOATING 2 OPERAND MULR.RG, PROD.$\overline{MG}$	PROD<Ø	PROD=Ø	FLOATING OVERFLOW	Ø
45FD	MULG3 MULTIPLY G_FLOATING 3 OPERAND MULR.RG, MULD.$\overline{RG}$, PROD.WG	PROD<Ø	PROD=Ø	FLOATING OVERFLOW	Ø
64FD	MULH2 MULTIPLE H_FLOATING 2 OPERAND MULR.RH, PROD.$\overline{MH}$	PROD<Ø	PROD=Ø	FLOATING OVERFLOW	Ø
65FD	MULH3 MULTIPLY H_FLOATING 3 OPERAND MULR.RH, MULD.$\overline{RH}$, PROD.WH	PROD<Ø	PROD=Ø	FLOATING OVERFLOW	Ø
C4	MULL2 MULTIPLY LONG 2 OPERAND MULR.RL, PROD.ML	PROD<Ø	PROD=Ø	INTEGER OVERFLOW	Ø
C5	MULL3 MULTIPLY LONG 3 OPERAND MULR.RL, MULD.RL, PROD.WL	PROD<Ø	PROD=Ø	INTEGER OVERFLOW	Ø
25	MULP MULTIPLY PACKED MULRLEN.RW, MULRADDR.AB, MULDLEN.RW, MULDADDR.AB, PRODLEN.RW, PRODADDR.AB	PROD STRING<Ø	PROD STRING=Ø	DECIMAL OVERFLOW	Ø
A4	MULW2 MULTIPLY WORD 2 OPERAND MULR.RW, PROD.MW	PROD<Ø	PROD=Ø	INTEGER OVERFLOW	Ø
A5	MULW3 MULTIPLY WORD 3 OPERAND MULR.RW, MULD.RW, PROD.WW	PROD<Ø	PROD=Ø	INTEGER OVERFLOW	Ø
Ø1	NOP NO OPERATION	N	Z	V	C
75	POLYD EVALUATE POLYNOMIAL DOUBLE ARG.RD, DEGREE.RW, TBLADDR.AB	RØ<Ø	RØ=Ø	FLOATING OVERFLOW	Ø
55	POLYF EVALUATE POLYNOMIAL FLOATING ARG.RF, DEGREE.RW, TBLADDR.AB	RØ<Ø	RØ=Ø	FLOATING OVERFLOW	Ø

Instructions		Condition Codes			
		N	Z	V	C
55FD	POLYG POLYNOMIAL EVALUATION G_FLOATING ARG.RG, DEGREE.RW, TBLADDR.AB	R0<0	RO=0	FLOATING OVERFLOW	0
75FD	POLYH POLYNOMIAL EVALUATION H_FLOATING ARG.RH, 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
7F	PUSHAG PUSH ADDRESS OF G_FLOATING SRC.AG	SRC<0	SRC=0	0	C
7FFD	PUSHAH PUSH ADDRESS H_FLOATING SRC.AH	SRC<0	SRC=0	0	C
DF	PUSHAL PUSH ADDRESS OF LONG SRC.AL	(SP)<0	(SP)=0	0	C
7FFD	PUSHAO PUSH ADDRESS OCTAWORD SRC.AO	SRC<0	SRC=0	0	C
7F	PUSHAQ PUSH ADDRESS OF QUAD SRC.AQ	(SP)<0	(SP)=0	0	C

Instructions		Condition Codes			
		N	Z	V	C
3F	PUSHAW PUSH ADDRESS OF WORD SRC.AW	(SP)<Ø	(SP)=Ø	Ø	C
DD	PUSHL PUSH LONG SRC.RL	SRC<Ø	SRC=Ø	Ø	C
BB	PUSHR PUSH REGISTERS MASK.RW	N	Z	V	C
Ø2	REI RETURN FROM EXCEPTION OR INTERRUPT	SAVED PSL <3>	SAVED PSL <2>	SAVED PSL <1>	SAVED PSL <Ø>
5E	REMQHI REMOVE ENTRY FROM QUEUE AT HEAD, INTERLOCKED HEADER.AQ, ADDR.WL				
	REMOVAL SUCCESSFUL:	Ø	(HEADER)=Ø !QUEUE EMPTY	TMP1=Ø !NO ENTRY TO REMOVE	Ø
	REMOVAL UNSUCCESSFUL:	Ø	Ø	1 !NOTHING REMOVED	1 !SECONDARY INTER-LOCK FAILED
5F	REMQTI REMOVE ENTRY FROM QUEUE AT TAIL, INTERLOCKED HEAD.AQ, ADDR.WL				
	REMOVAL SUCCESSFUL:	Ø	(HEADER+4)=Ø !QUEUE EMPTY	TMP3=Ø !NO ENTRY TO REMOVE	Ø
	REMOVAL UNSUCCESSFUL:	Ø	Ø	1 !NOTHING REMOVED	1 !SECONDARY INTER-LOCK FAILED
ØF	REMQUE REMOVE FORM QUEUE ENTRY.AB, ADDR.WL	ENTRY<ENTRY+4	ENTRY=ENTRY+4	ENTRY=ENTRY+4	ENTRY LSSU ENTRY+4
Ø4	RET RETURN FROM CALLED PROCEDURE	TMPL<3>	TMPL<2>	TMPL<1>	TMPL<Ø>
9C	ROTL ROTATE LONG CNT.RB, SCR.RL, DST.WL	DST<Ø	DST=Ø	Ø	C
Ø5	RSB RETURN FROM SUBROUTINE	N	Z	V	C

Instructions	Condition Codes			
	N	Z	V	C
D9 SBWC SUBTRACT WITH CARRY SUB.RL, DIF.ML	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW FROM MSB
2A SCANC SCAN FOR CHARACTER LEN.RW, ADDR.AB, TBLADDR.AB, MASK.RB	Ø	RØ=Ø	Ø	Ø
3B SKPC SKIP CHARACTER CHAR.RB, LEN.RW, ADDR.AB	Ø	RØ=Ø	Ø	Ø
F4 SOBGEQ SUBTRACT ONE AND BRANCH ON GREATER OR EQUAL INDEX.ML, DISPL.BB	INDEX<Ø	INDEX=Ø	INTEGER OVERFLOW	C
F5 SOBGTR SUBTRACT ONE AND BRANCH ON GREATER INDEX.ML, DISPL.BB	INDEX<Ø	INDEX=Ø	INTEGER OVERFLOW	C
2B SPANC SPAN CHARACTERS LEN.RW, ADDR.AB, TBLADDR.AB, MASK.RB	Ø	RØ=Ø	Ø	Ø
82 SUBB2 SUBTRACT BYTE 2 OPERAND SUB.RB, DIF.MB	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW FROM MSB
83 SUBB3 SUBTRACT BYTE 3 OPERAND SUB.RB, MIN.RB, DIF.WB	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW FROM MSB
62 SUBD2 SUBTRACT D_FLOATING 2 OPERAND SUB.RD, DIF.MD	DIFF<Ø	DIFF=Ø	FLOATING OVERFLOW	Ø
63 SUBD3 SUBTRACT D_FLOATING 3 OPERAND SUB.RD, MIN.RD, DIF.WD	DIFF<Ø	DIFF=Ø	FLOATING OVERFLOW	Ø
42 SUBF2 SUBTRACT F_FLOATING 2 OPERAND SUB.RF, DIF.MF	DIFF<Ø	DIFF=Ø	FLOATING OVERFLOW	Ø
43 SUBF3 SUBTRACT F_FLOATING 3 OPERAND SUB.RF, MIN.RF, DIF.WF	DIFF<Ø	DIFF=Ø	FLOATING OVERFLOW	Ø
42FD SUBG2 SUBRTACT G_FLOATING 2 OPERAND SUB.RG, DIF.MG	DIF<Ø	DIF=Ø	FLOATING OVERFLOW	Ø
43FD SUBG3 SUBTRACT G_FLOATING 2 OPERAND SUB.RG, MIN.RG, DIF.WG	DIF<Ø	DIF=Ø	FLOATING OVERFLOW	Ø

Instructions		Condition Codes			
		N	Z	V	C
62FD	SUBH2 SUBTRACT H_FLOATING 2 OPERAND SUB.RH, DIF.MH	DIF<Ø	DIF=Ø	FLOATING OVERFLOW	Ø
63FD	SUBH3 SUBTRACT H_FLOATING 3 OPERAND SUB.RH, MIN.RH, DIF.WH	DIF<Ø	DIF=Ø	FLOATING OVERFLOW	Ø
C2	SUBL2 SUBTRACT LONG 2 OPERAND SUB.RL, DIF.ML	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW FROM MSB
C3	SUBL3 SUBTRACT LONG 3 OPERAND SUB.RL, MIN.RL, DIF.WL	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW FROM MSB
22	SUBP4 SUBTRACT PACKED 4 OPERAND SUBLEN.RW, SUBADDR.AB, DIFLEN.RW, DIFADDR.AB	DIFF STRING<Ø	DIFF STRING=Ø	DECIMAL OVERFLOW	Ø
23	SUBP6 SUBTRACT NUMERPACKED 6 OPERAND SUBLEN.RW, SUBADDR.AB, MINLEN.RW, MINADDR.AB, DIFLEN.RW, DIFADDR.AB	DIFF STRING<Ø	DIFF STRING=Ø	DECIMAL OVERFLOW	Ø
A2	SUBW2 SUBTRACT WORD 2 OPERAND SUB.RW, DIF.MW	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW FROM MSB
A3	SUBW3 SUBTRACT WORD 3 OPERAND SUB.RW, MIN.RW, DIF.WW	DIFF<Ø	DIFF=Ø	INTEGER OVERFLOW	BORROW FROM MSB
Ø7	SVPCTX SAVE PROCESS CONTEXT	N	Z	V	C
95	TSTB TEST BYTE SRC.RB	SRC<Ø	SRC=Ø	Ø	Ø
73	TSTD TEST D_FLOATING SRC.RD	SRC<Ø	SRC=Ø	Ø	Ø
53	TSTF TEST F_FLOATING SRC.RF	SRC<Ø	SRC=Ø	Ø	Ø
53FD	TSTG TEST G_FLOATING SRC.RG	SRC<Ø	SRC=Ø	Ø	Ø
73FD	TSTH TEST H_FLOATING SRC.RH	SRC<Ø	SRC=Ø	Ø	Ø

Instructions		Condition Codes			
		N	Z	V	C
D5	TSTL TEST LONG SRC.RL	SRC<Ø	SRC=Ø	Ø	Ø
B5	TSTW TEST WORD SRC.RW	SRC<Ø	SRC=Ø	Ø	Ø
FC	XFC EXTENDED FUNCTION CALL	Ø	Ø	Ø	Ø
8C	XORB2 EXCLUSIVE-OR BYTE 2 OPERAND MASK.RB, DST.MB	DST<Ø	DST=Ø	Ø	C
8D	XORB3 EXCLUSIVE-OR BYTE 3 OPERAND MASK.RB, SRC.RB, DST.WB	DST<Ø	DST=Ø	Ø	C
CC	XORL2 EXCLUSIVE-OR LONG 2 OPERAND MASK.RL, DST.ML	DST<Ø	DST=Ø	Ø	C
CD	XORL3 EXCLUSIVE-OR LONG 3 OPERAND MASK.RL, SRC.RL, DST.WL	DST<Ø	DST=Ø	Ø	C
AC	XORW2 EXCLUSIVE-OR WORD 2 OPERAND MASK.RW, DST.MW	DST<Ø	DST=Ø	Ø	C
AD	XORW3 EXCLUSIVE-OR WORD 3 OPERAND MASK.RW, SRC.RW, DST.WW	DST<Ø	DST=Ø	Ø	C
57	* RESERVED TO DIGITAL *				
59	* RESERVED TO DIGITAL *				
5A	* RESERVED TO DIGITAL *				
5B	* RESERVED TO DIGITAL *				
5C	* RESERVED TO DIGITAL *				
5D	* RESERVED TO DIGITAL *				
5E	* RESERVED TO DIGITAL *				
5F	* RESERVED TO DIGITAL *				
77	* RESERVED TO DIGITAL *				

Instructions			Condition Codes			
			N	Z	V	C
FD	ESCD	* RESERVED TO DIGITAL *				
FE	ESCE	* RESERVED TO DIGITAL *				
FF	ESCF	* RESERVED TO DIGITAL *				
1ØØ		* RESERVED TO DIGITAL *				
31FD		* RESERVED TO DIGITAL *				
57FD		* RESERVED TO DIGITAL *				
58FD		* RESERVED TO DIGITAL *				
59FD		* RESERVED TO DIGITAL *				
5AFD		* RESERVED TO DIGITAL *				
5BFD		* RESERVED TO DIGITAL *				
5CFD		* RESERVED TO DIGITAL *				
5DFD		* RESERVED TO DIGITAL *				
5EFD		* RESERVED TO DIGITAL *				
5FFD		* RESERVED TO DIGITAL *				
77FD		* RESERVED TO DIGITAL *				
78FD		* RESERVED TO DIGITAL *				
79FD		* RESERVED TO DIGITAL *				
7AFD		* RESERVED TO DIGITAL *				
7BFD		* RESERVED TO DIGITAL *				
8ØFD		* RESERVED TO DIGITAL *				
97FD		* RESERVED TO DIGITAL *				
9AFD		* RESERVED TO DIGITAL *				

Instructions		Condition Codes			
		N	Z	V	C
F5FD	* RESERVED TO DIGITAL *				
F8FD	* RESERVED TO DIGITAL *				
FCFF	* RESERVED TO DIGITAL *				
FFFF	RESERVED FOR ALL TIME				

BRANCH CONDITIONS

Opcode	Conditions
BGTR	N or Z = Ø
BLEQ	N or Z = 1
BNEQ	Z = Ø
BNEQU	Z = Ø
BEQL	Z = 1
BEQLU	Z = 1
BGEQ	N = Ø
BLSS	N = 1
BGTRU	C or Z = Ø
BLEQU	C or Z = 1
BVC	V = Ø
BVS	V = 1
BGEQU	C = Ø
BCC	C = Ø
BLSSU	C = 1
BCS	

CHAPTER 3

EXCEPTIONS AND INTERRUPTS

INTERRUPT AND EXCEPTION DEFINITIONS

Interrupt	An event other than an exception, branch, jump, case, or call instruction that changes the normal flow of instruction execution. Interrupts are generally external to the process executing when the interrupt occurs.
Exception	An event detected by hardware other than an interrupt, jump, branch, case, or call instruction that changes the normal flow of instruction execution. An exception is always caused by the execution of an instruction or set of instructions. There are three types of exceptions.
• Trap	An exception condition that occurs at the end of the instruction that caused the exception. The PC saved on the stack is the address of the next instruction that would normally have been executed.
• Fault	A condition that occurs in the middle of an instruction and leaves the registers and memory in a consistent state that allows the instruction to restart and give correct results once the fault has been cleared or eliminated.
• Abort	An exception that occurs in the middle of an instruction and leaves the registers and memory in an indeterminate state that may prohibit an instruction restart.

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	CHME	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.

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	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

* VAX-11/780 only.

EXCEPTION CONDITIONS

Condition	Vector
Machine check	04
Kernel stack not valid	08
Reserved DEC opcodes and 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
Compatibility mode trap	30
Arithmetic trap	34
CHMK opcode	40
CHME opcode	44
CHMS opcode	48
CHMU opcode	4C

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

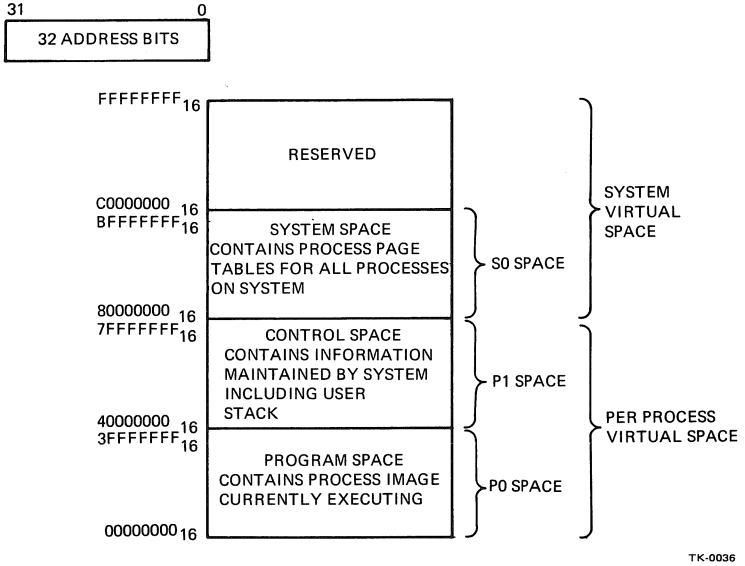
PROCESS CONTROL BLOCK (PCB)

			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

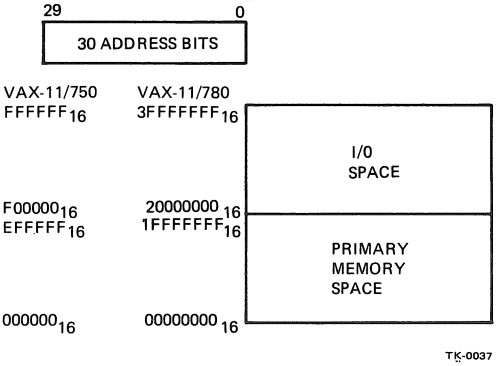
CHAPTER 4
MEMORY MANAGEMENT

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)

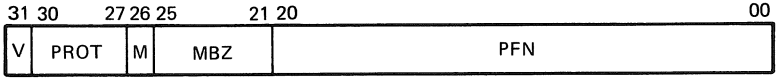
PROGRAM REGION LENGTH REGISTER
(CONTAINS THE NUMBER OF PAGE TABLE ENTRIES, N)

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

PAGE TABLE ENTRY PROTECTION CODES

Code	Meaning			
	K	E	S	U
Ø Ø Ø Ø	*	*	*	*
Ø Ø Ø 1	Unpredictable			
Ø Ø 1 Ø	R/W	*	*	*
Ø Ø 1 1	RO	*	*	*
Ø 1 Ø Ø	R/W	R/W	R/W	R/W
Ø 1 Ø 1	R/W	R/W	*	*
Ø 1 1 Ø	R/W	RO	*	*
Ø 1 1 1	RO	RO	*	*
1 Ø Ø Ø	R/W	R/W	R/W	*
1 Ø Ø 1	R/W	R/W	RO	*
1 Ø 1 Ø	R/W	RO	RO	*
1 Ø 1 1	RO	RO	RO	*
1 1 Ø Ø	R/W	R/W	R/W	RO
1 1 Ø 1	R/W	R/W	RO	RO
1 1 1 Ø	R/W	RO	RO	RO
1 1 1 1	RO	RO	RO	RO

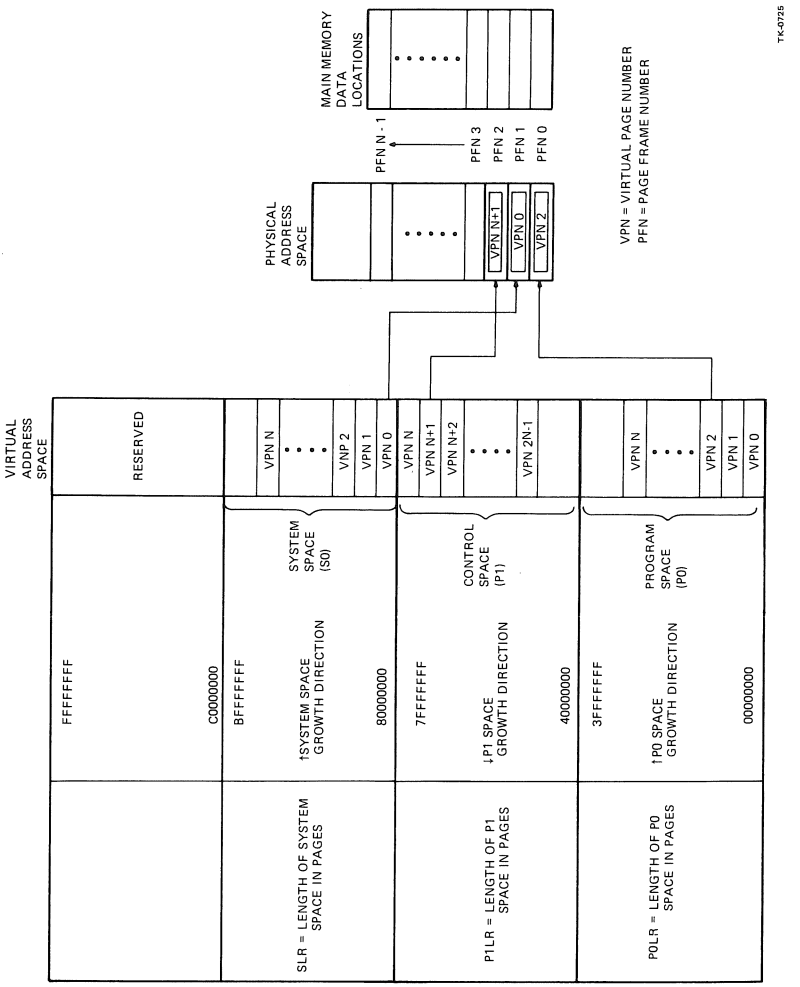
K Kernel
E Executive
S Supervisor
U User

* No access
RØ Read only
R/W Read write

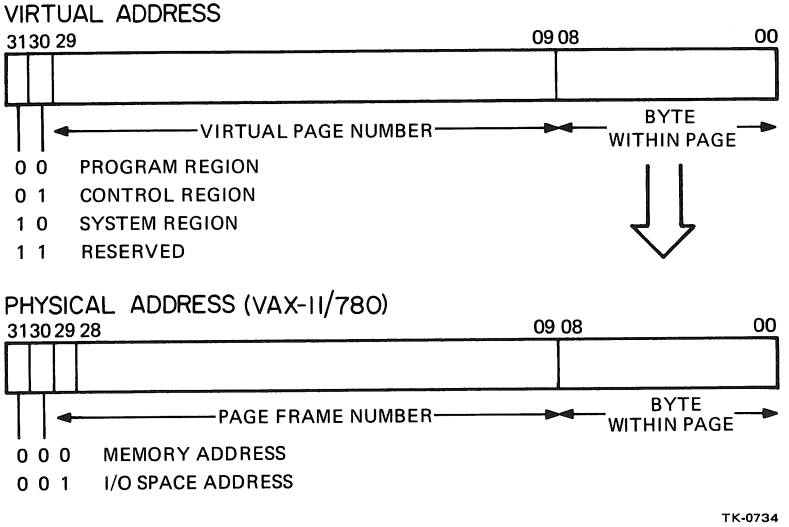
Mode

Ø (ØØ) Kernel
1 (Ø1) Executive
2 (1Ø) Supervisor
3 (11) User

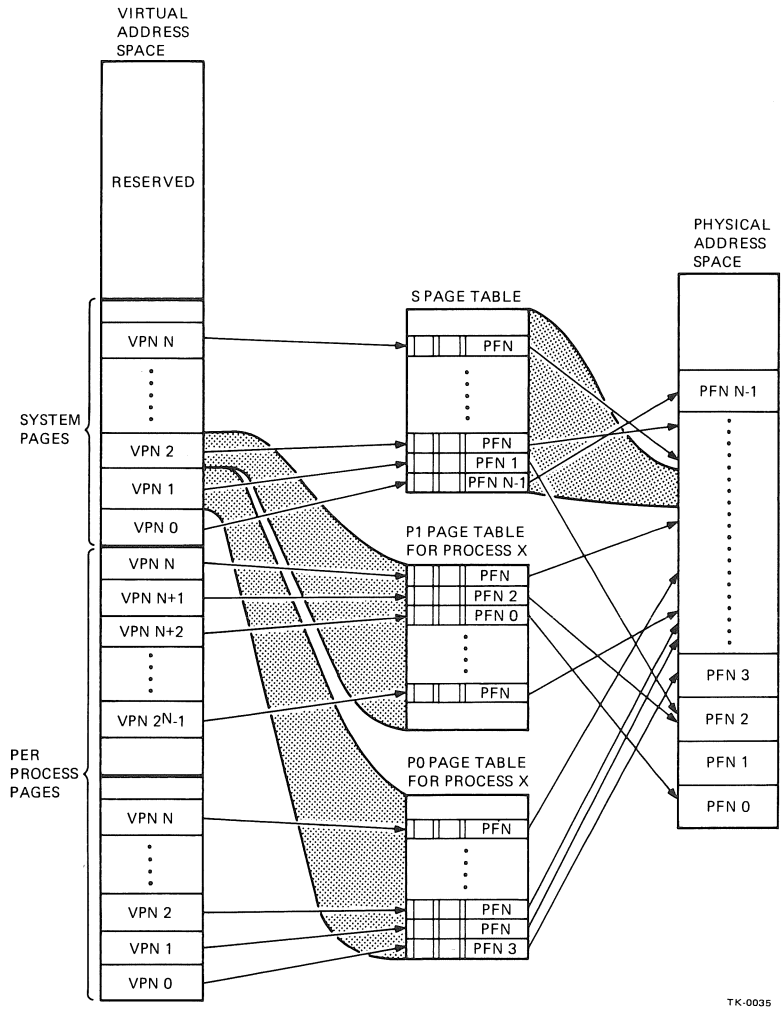
EXAMPLE OF PAGE FRAME ALLOCATION (RELOCATION)



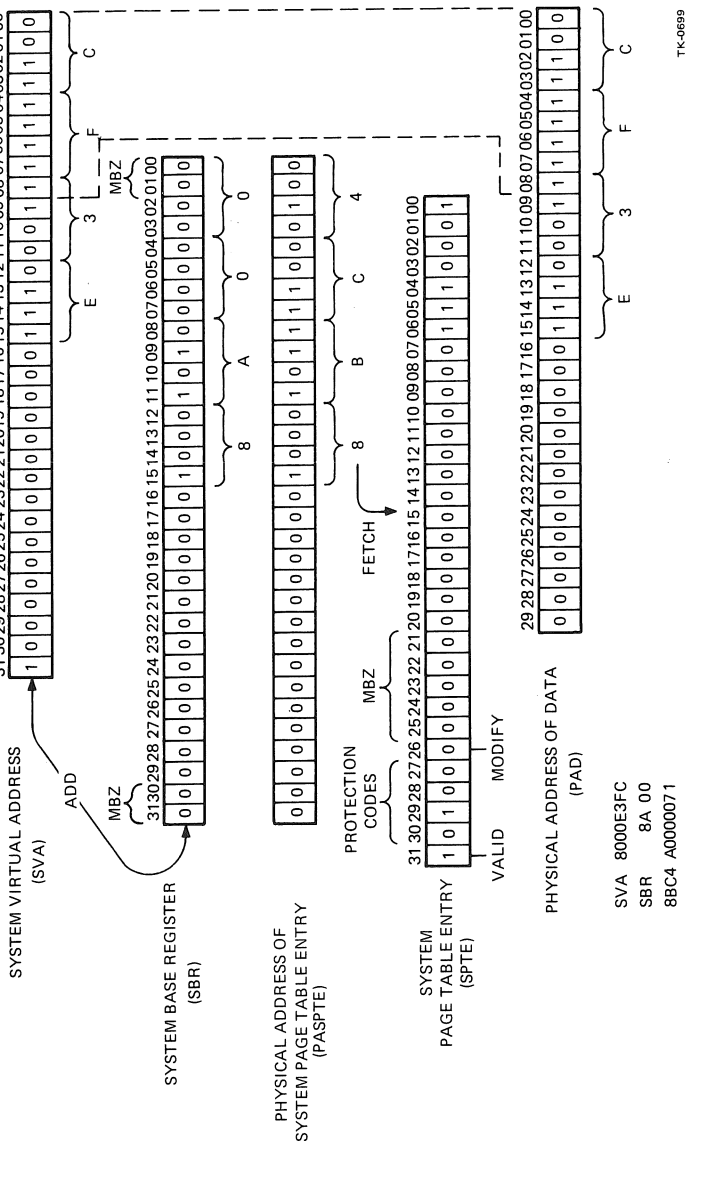
VIRTUAL AND PHYSICAL ADDRESS FORMATS



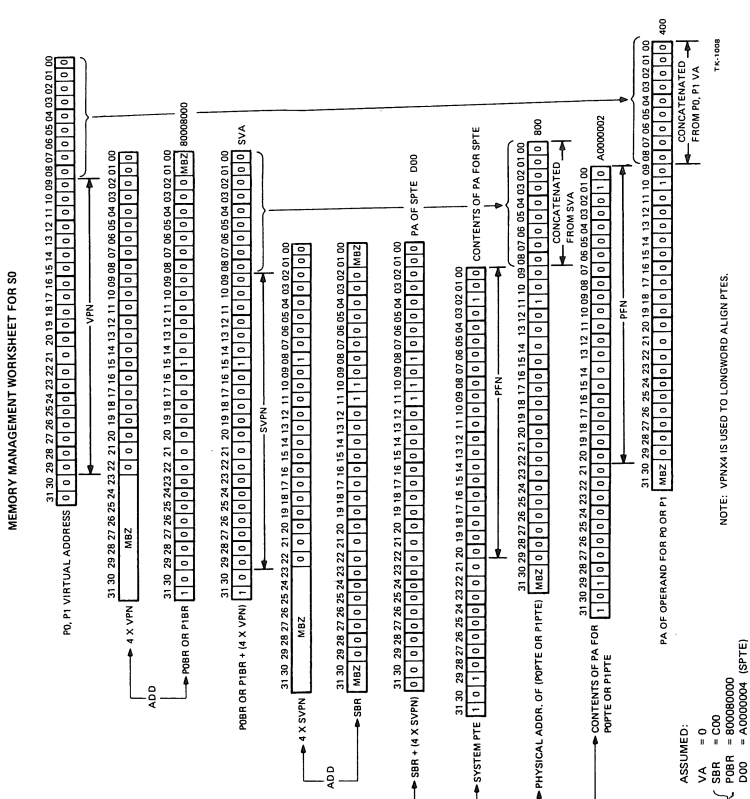
VIRTUAL PAGES MAPPED TO PHYSICAL SPACE



SYSTEM VIRTUAL-TO-PHYSICAL ADDRESS TRANSLATION, EXAMPLE



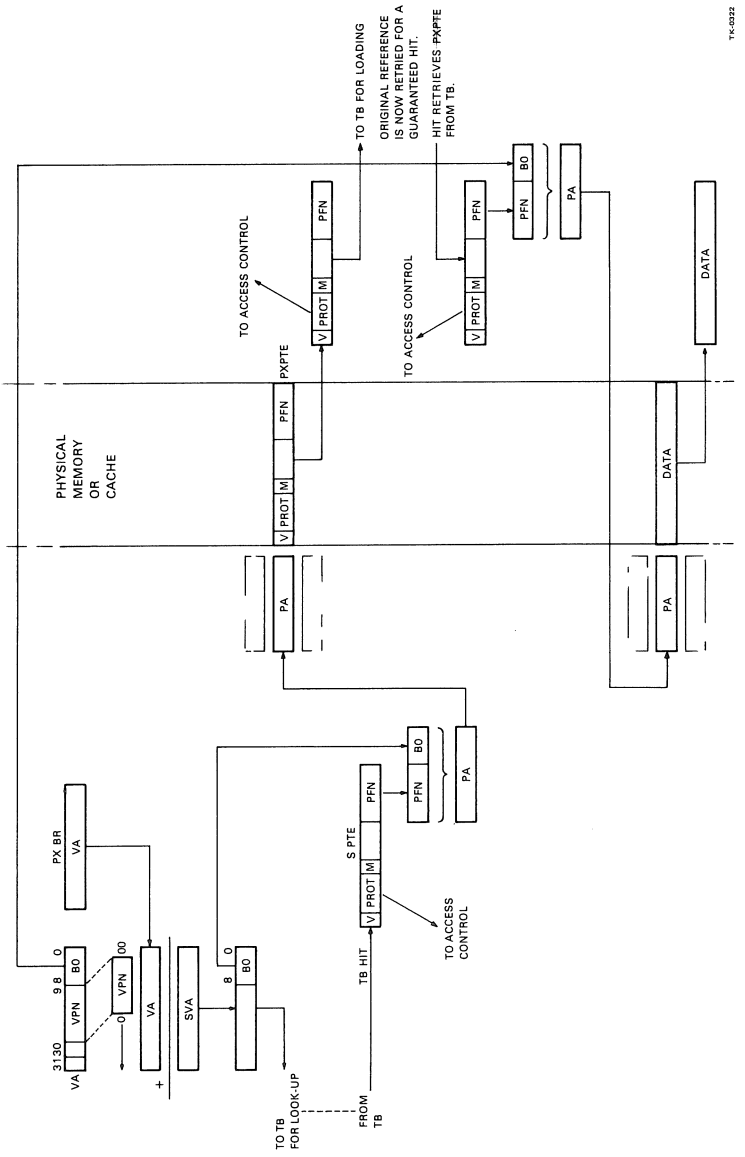
PROCESS VIRTUAL-TO-PHYSICAL ADDRESS TRANSLATION, EXAMPLE



See page 141 for flows

page 56

ADDRESS CALCULATION FOR A TB HIT DURING A MISS MICROTRAP



CHAPTER 5
OPERATING SYSTEM

VMB is loaded into physical memory and gains control via a JMP instruction from CONSOLE, boot block Ø on the boot device, or BOOT58 in the routine START_BOOT.

VMB begins by creating and initializing an SCB. If the software bootstrap control flags specified a bootstrap breakpoint, VMB then executes a BPT instruction that transfers control to XDELTA.

After the XDELTA breakpoint, VMB initializes a system data structure, that is, a restart parameter block (RPB) that allows a system reboot after a power failure or crash. The RPB holds the bootstrap input registers, the boot device's CSR and bus configuration register (CR), the address of the RPB, and pointers to a primitive device driver.

VMB, the primary bootstrap, also identifies all physical memory in the configuration by creating a bit map in which each bit represents one page of physical memory. In the process of testing all memory, VMB determines which NEXUSes on the system bus are attached to adapters. For every adapter present, VMB records the adapter type in the RPB.

Finally, VMB chooses a secondary bootstrap image - either by default, by boot flag settings, or by soliciting a file specification from the user. VMB uses a minimal driver for the bootstrap device to load the secondary image into memory and transfers control to that bootstrap.

The secondary bootstrap - usually SYSBOOT.EXE - uses the minimal driver from VMB to read and write to and from the bootstrap device. Thus, SYSBOOT is device independent.

VMB has CPU dependencies such as system bus addresses, memory controller registers, and bus adapter register formats. Therefore, VMB consists of common code that applies to all VAX implementations, and CPU-specific code that applies to one hardware implementation only. The current version supports the following CPUs:

- VAX-11/78Ø
- VAX-11/75Ø
- VAX-11/73Ø

Inputs:

R0 <31:4> MBZ
<3:0> boot device type code

Ø	MASSBUS device (RMØ2/3, RPØ4/5/6/7)
1	RKØ6/7
2	RLØ1/2
32	RSC on CI
64	Console block storage device

R1 Boot device's bus address

11/78Ø	<31:4> MBZ <3:Ø> TR number of adapter

11/750 <31:24> MBZ
<23:0> Address of the I/O page for the boot device's adapter

R2 UNIBUS:

<31:18> MBZ
<17:0> UNIBUS address of the device's CSR

MASSBUS:

<31:4> MBZ
<3:0> Adapter's controller/formatter number

CI:

<31:8> MBZ
<7:0> HSC port number (station address)

R3 Boot device unit number

R5 Software boot control flags

Bit	Meaning
0	RPB$V_CONV. Conversational boot. At various points in the system boot procedure, the bootstrap code solicits parameters and other input from the console terminal.
1	RPB$V_DEBUG. Debug. If this flag is set, VMS maps the code for the XDELTA debugger into the system page tables of the running system.
2	RPB$V_INIBPT. Initial breakpoint. If RPB$V_DEBUG is set, VMS executes a BPT instruction immediately after enabling mapping.
3	RPB$V_BLOCK. Secondary boot from boot block. Secondary bootstrap is a single 512-byte block, whose LBN is specified in R4.
4	RPB$V_DIAG. Diagnostic boot. Secondary bootstrap is image called [SYSMAINT]DIAGBOOT.EXE.
5	RPB$V_BOOBPT. Bootstrap breakpoint. Stops the primary and secondary bootstraps with a breakpoint instruction before testing memory.

Bit	Meaning
6	RPB$V_HEADER. Image header. Takes the transfer address of the secondary bootstrap image from that file's image header. If RPB$V_HEADER is not set, transfers control to the first byte of the secondary boot file.
7	RPB$V_NOTEST. Memory test inhibit. Sets a bit in the PRN bit map for each page of memory present. Does not test the memory.
8	RPB$V_SOLICT. File name. VMB prompts for the name of a secondary bootstrap file.
9	RPB$V_HALT. Halt before transfer. Executes a HALT instruction before transferring control to the secondary bootstrap.
10	RPB$V_NOPFND. No PFN deletion (not implemented; intended to tell VMB not to read a file from the boot device that identifies bad or reserved memory pages, so that VMB does not mark these pages as valid in the PFN bitmap).
11	RPB$V_MPM. Specifies that multiport memory is to be used for the total executive memory requirement. No local memory is to be used. This is for tightly coupled multiprocessing.
12	RPB$V_USEMPM. Specifies that multiport memory should be used in addition to local memory, as though both were one single pool of pages.
13	RPB$V_MEMTEST Specifies that a more extensive algorithm be used when testing main memory for hardware uncorrectable (RDS) errors.
<31:28>	RPB$V_TOPSYS Specifies the top level directory number for system disks with multiple systems.

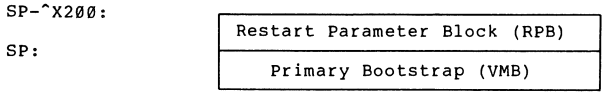
The hardware or the CONSOLE program sets up the next three registers after a system crash or power failure:

```
R10       - halt PC
R11       - halt PSL
AP        - halt code

SP        - <base_address + ^X200> of 64KB of good memory
```

Implicit inputs:

When VMB gains control, physical memory looks like the following diagram:

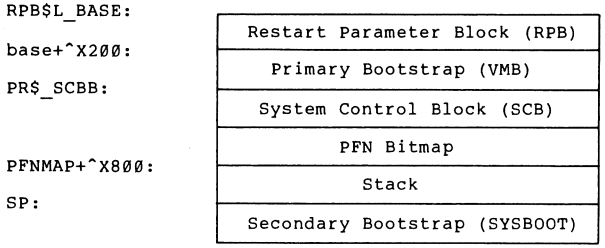


Outputs:

```
R10       - base address of region containing secondary
            bootstrap
R11       - address of restart parameter block
SP        - current stack pointer
PR$_SCBB  - system control block address
```

Implicit outputs:

When VMB transfers control to the secondary bootstrap, physical memory is laid out as in the following diagram:



The design for the PFN bitmap has been extended to handle more than four pages of bitmap equal to eight megabytes of memory. Bitmaps that do not fit in the four-page reserved area are now allocated contiguous good pages in higher memory. Assuming that the pages are actually good, the bitmap is placed at the RPB address plus one megabyte. If a page in that area is bad, the next contiguous run of pages that is big enough is where the bitmap will be placed. For backward compatibility, the RPB$Q_PFNMAP descriptor in the RPB points at the preallocated four-page bitmap, which is correct for the low eight megabytes of memory. The real descriptor for the bitmap is passed in the argument list.

Condition Name/Type	Explanation	Additional Arguments
SS$_ACCVIO (Fault)	Access Violation	1. Reason for access violation. This is a mask with the format: Bit Ø = type of access violation Ø = page table entry protection code did not permit intended access 1 = PØLR, P1LR, or SLR length violation Bit 1 = page table entry reference Ø = specified virtual address not accessible 1 = associated page table entry not accessible Bit 2 = intended access Ø = read 1 = modify 2. Virtual address to which access was attempted
SS$_ARTRES (Trap)	Reserved arithmetic trap	None
SS$_ASTFLT (Fault)	Stack invalid during attempt to deliver an AST	1. Stack pointer value when fault occurred 2. AST parameter of failed AST 3. Program counter (PC) at AST delivery interrupt 4. Processor status longword (PSL) at AST delivery interrupt* 5. Program counter (PC) to which AST would have been delivered* 6. processor status longword (PSL) to which AST would have been delivered*
SS$_BREAK (Fault)	Breakpoint instruction encountered	None

* The PC and PSL normally included in the signal array are not included in this argument list. The stack pointer of the access mode receiving this exception is reset to its initial value.

Condition Name/Type	Explanation	Additional Arguments
SS$_CMODSUPR (Trap)	Change mode to supervisor instruction encountered +	Change mode code. The possible values are -32768 through 32767.
SS$_CMODUSER (Trap)	Change mode to user instruction encountered +	Change mode code. The possible values are -32768 through 32767.
SS$_COMPAT (Fault)	Compatibility mode exception. This exception condition can only occur when executing in compatibility mode.**	Type of compatibility exception. The possible values are: Ø = Reserved instruction execution 1 = BPT instruction executed 2 = IOT instruction executed 3 = EMT instruction executed 4 = TRAP instruction executed 5 = illegal instruction executed 6 = Odd address fault 7 = TBIT trap
SS$_DECOVF (Trap)	Decimal overflow	None
SS$_FLTDIV (Trap)	Floating/decimal divide by zero	None
SS$_FLTDIV_F (Fault)	Floating overflow	None
SS$_FLTOVF_F (Fault)	Floating overflow fault	None
SS$_FLTUND (Trap)	Floating underflow	None
SS$_FLTUND_F (Fault)	Floating underflow fault	None

+ If a change mode handler has been declared for user or supervisor modes with the declare change mode or compatibility mode handler ($DCLCMH) system service, that routine receives control when the associated trap occurs.

** If a compatibility mode handler has been declared with the declare change mode or compatibility mode handler ($DCLCMH) system service, that routine receives control when this fault occurs.

Condition Name/Type	Explanation	Additional Arguments
SS$_INTDIV (Trap)	Integer divide by zero	None
SS$_INTOVF (Trap)	Integer overflow	None
SS$_OPCCUS (Fault)	Opcode reserved to customer fault	None
SS$_OPCDEC (Fault)	Opcode reserved to DIGITAL fault	None
SS$_PAGRDERR (Fault)	Read error occurred during an attempt to read a faulted page from disk	1. Translation not valid reason. This is a mask with the format: Bit Ø = Ø Bit 1 = page table entry reference Ø = specified virtual address not valid 1 = associated page table entry not valid Bit 2 = intended access Ø = read 1 = modify
SS$_RADRMOD (Fault)	Attempt to use a reserved addressing mode	None
SS$_ROPRAND (Fault)	Attempt to use a reserved operand	None
SS$_SSFAIL (Fault)	System service failure (when system service failure exception mode is enabled)	Status return from system service (RØ) (The same value is in RØ of the mechanism array)
SS$_SUBRNG	Subscript range trap	None
SS$_TBIT (Fault)	Trace bit is pending following an instruction	None

PROCESS PRIVILEGES

Privilege	Definition
CMKRNL	May change mode to kernel.
CMEXEC	May change mode to executive.
SYSNAM	May insert in system logical name table.
GRPNAM	May insert in group logical name table.
ALLSPOOL	May allocate spooled device.
DETACH	May create detached processes.
DIAGNOSE	May diagnose devices.
LOG_IO	May do logical I/O.
GROUP	May affect other processes in same group.
ACNT	May suppress accounting message.
PRMCER	May create permanent common event cluster.
PRMMBX	May create permanent mailbox.
PSWAPM	May change process swap mode.
ALTPRI	May set any priority value.
SETPRV	May set any privilege bit.
TMPMBX	May create temporary mailbox.
WORLD	May affect other processes in the world.
OPER	May perform operator functions.
EXQUOTA	May exceed quota.
NETMBX	May create network device.
VOLPRO	May override volume protection.
PHY_IO	May do physical I/O.
BUGCHK	May make bug check log entries.
PRMGBL	May create permanent global sections.
SYSGBL	May create system-wide global sections.
MOUNT	May execute mount ACP functions.
PFNMAP	May map to specific physical pages.
SHMEM	May create global sections and mailboxes in multiport memory.
SYSPRV	May access objects via system protection.
SYSLCK	May allow user's process to lock system-wide resources.

SYSTEM ERROR LOG FORMATTER (SYE) UTILITY

The SYE utility is a system management tool that selectively reports the contents of an error log file. The VAX/VMS system automatically writes messages to the latest version of an error log file named SYS$ERRORLOG:ERRLOG.SYS whenever one of the following events is detected:

- Errors - Device errors, machine checks, bus errors, soft error correcting code (ECC) errors, asynchronous write errors, or hard ECC errors
- Configuration changes - Volume mounts and dismounts
- System events - Cold startup, warm startup, crash startup, message received from the send message to error logger ($SNDERR) system service, or a time stamp

All SYE reports are 72 columns wide, so they can be displayed at the terminal. SYE reports are primarily intended to assist DIGITAL field service personnel. In some cases, however, they can assist in system management by identifying recurrent failures that indicate outside attention is required.

Additional details about error logging can be found in the VAX/VMS System Management and Operations Guide.

INVOKING AND TERMINATING SYE

The following command invokes the SYE utility:

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

Only users with the SYSPRV privilege or a system UIC can access the error log file.

SYE normally runs to termination and issues a successful completion message on the device SYS$OUTPUT. To request additional SYE reports, you must reinvoke the utility. You can interrupt SYE at any time with CTRL/Y.

SYE PROMPTS

SYE displays a series of prompts to which you respond with a value or take the default by pressing CTRL/Z or the RETURN key. The prompt ends with a question mark and indicates the default within square brackets. A list of prompts and types of valid responses follows. The responses are described in later sections.

```
Prompt                                          Valid Response

_INPUT FILE   [SYS$ERRORLOG:ERRLOG.OLD]         Input-file-spec
_OUTPUT FILE  [SYS$OUTPUT]                      Output-file-spec
_OPTIONS      [ROLL-UP]                         Report-option
_DEVICE NAME  [<CR>]                            Category-type
_AFTER DATE   [FIRST ENTRY]                     After-date-spec
_BEFORE DATE  [LAST ENTRY]                      Before-date-spec
_
```

VALID RESPONSES TO SYE PROMPTS

This section describes the responses you can enter to each of the SYE prompts in the proper order.

Input-file-spec

The file specification for the input log file. If you do not specify a file, but simply respond by pressing RETURN, SYE uses the highest version of SYS$ERRORLOG:ERRLOG.OLD by default.

The file specification may be any valid VAX/VMS file specification. Wild card characters and logical names are allowed. You may also use the special file specification MAILBOX, which reports error log entries as they occur on the system, in the form you request. However, use of the MAILBOX feature of SYE requires the user privilege DIAGNOSE. The fields you omit default to those for VAX-11 FORTRAN unit 1, that is, the highest version of SYS$SYSROOT: [default-directory]FOR001.DAT (SYE is written in FORTRAN).

Output-file-spec

The file specification for the output report file; the default is SYS$OUTPUT, which is usually your terminal.

If you want to direct the output to a specific file, simply identify it with any valid file specification. Logical names are allowed. Omitted fields default to those for VAX-11 FORTRAN unit 2, that is, the highest version of SYS$SYSROOT: [default-directory]FOR002.DAT.

Report-option

Any one of the following single-character abbreviations to request the type of report you want, or RETURN (to accept the default option, which is the rollup report).

B (brief) - A brief report of each error, configuration change, and system event being reported.

C (cryptic) - A report of the contents of the device registers at the time a particular device error occurred. The values are represented in hexadecimal with no explanations. This report is only meaningful when errors on a single device are requested.

R (rollup) - A report providing totals for each category of error report requested (either a single category or all categories can be requested). This is the default.

S (standard) - A full report of each error, configuration change, and system event being reported.

Category-type

Either the name of a device whose device errors you want reported or else a qualifier that requests particular categories of errors that you want to review. Details follow on how to specify these. If you respond by pressing RETURN, you receive a report of all errors (for every category and every device).

Device name - Valid VAX/VMS-supported device names appear in the device name table in the VAX/VMS Command Language User's Guide. The asterisk (*) wild card character is allowed. For example, if you specify the device name DR*, you obtain a report of errors logged on all DR devices; if you specify DRA*, you obtain a report of errors on controller A of your DR device; a complete specification of DRA7 extracts errors only for the particular DR device on unit 7 of controller A.

This is a change from earlier versions of SYE. Previously you could request reports on all DR devices by specifying DR. Now you must specify DR* to obtain the same report. You should not specify an underscore (_) with the device name; however, you can specify a colon (:).

In addition, you can specify other DIGITAL-supported devices that may be on your system that were not provided with the VAX/VMS system.

SYE accepts a minus sign (-) as a prefix for a device name. That is, if you specify -DR_, the SYE report covers all devices except the DR disks.

Category type qualifiers - You can specify any one of the following category type qualifiers. Note that several of these qualifiers accept a device-name specification. For these qualifiers, you can specify one or more device-names, as just described. Specify multiple device-names in parentheses, separated by commas. Use a single asterisk (*) wild card character as the device-name specification to request a report of all the errors of that category on all the devices. (You can also use the asterisk wild card character in the device specification, as already described.)

/BU[=NO_ORS]	All bugcheck entries. You can request that the operator-requested shutdowns be omitted by specifying /BU=NO_ORS.
/CO=(device-name[,...])	Configuration changes due to volume mounts and volume dismounts.
/CP	Machine checks, applicable bus errors, applicable interrupts, and uninitialized SCB vector interrupts (on a VAX-11/780 only).

/DA=(device-name[,...])	Device attention entries. These device errors can occur even when there is no I/O request outstanding on the device. For example, such errors can occur when diagnostics are run on a TU78.
/DE=(device-name[,...])	Device error bit(s) set entries.
/DT=(device-name[,...])	Device I/O timeouts.
/ME	Memory errors detected by the scanning code and interrupts and fatal memory errors logged by the machine check code.
/SY	System events, such as system startups, system power fails, new error log file creation, system power fail restart, time stamps, operator messages, network messages, and receipt of messages from the $SNDERR system service.
/UN	Unknown device errors. This qualifier requests a report of errors found on all non-DIGITAL devices and all drivers not supported by DIGITAL. It also covers DIGITAL-supported drivers that may have been released since the last release of SYE. When you request information on the unknown devices, the report is developed on a "best try" basis, because SYE cannnot fully interpret errors on these devices.

You can also specify a minus sign (-) as a prefix for any of the qualifiers to indicate you want all the errors in every category except this one. However, you cannot combine the minus sign on the qualifier with a minus sign on one or more of its device names.

After-date-spec

Time on error log after which reporting begins, in the following format:

```
dd-mmm-yyyy hh:mm:ss.cc
```

This is almost the format given in the VAX/VMS Command Language User's Guide for the absolute time format. The only difference is you cannot specify a colon (:) between the year field (yyyy) and the hour field (hh). If you omit the entire time specification by pressing RETURN, SYE starts with the entries at the beginning of the log. You can omit portions of the time specification and obtain certain defaults. Refer to the description of the $BINTIM system service in the VAX/VMS System Service Reference Manual for the rules and examples of the defaults.

Before-date-spec

Time on error log before which reporting ends, in the following format:

```
dd-mmm-yyyy hh:mm:ss.cc
```

This is almost the format given in the VAX/VMS Command Language User's Guide for the absolute time format. The only difference is you cannot specify a colon (:) between the year field (yyyy) and the hour field (hh). If you omit the entire time specification by pressing RETURN, SYE includes all entries until the end of the log is reached. You can omit portions of the time specification and obtain certain defaults. Refer to the description of the $BINTIM system service in the VAX/VMS System Service Reference Manual for the rules and examples of the defaults.

ERROR MESSAGES

SYE rarely issues error messages on incorrect input; most often it simply reprompts. A few error messages originated by SYE have a facility code of SYE or some other system component such as RMS or SYSTEM. The VAX/VMS System Message and Recovery Procedures Manual lists these messages and provides explanations and suggested user actions. If any other type of error message appears in the report, you should rerun SYE to eliminate the error. Most SYE error messages are reported as VAX-11 FORTRAN messages. If you receive a recurring error message that does not disappear when you rerun SYE, you should submit a Software Performance Report (SPR) to DIGITAL. (SPRs are described in the VAX/VMS System Management and Operations Guide.)

TYPICAL SYE UTILITY ENTRY

```
$ SET DEFAULT SYS$ERRORLOG
$ RENAME ERRLOG.SYS ERRLOG.OLD/NEW_VERSION
$ RUN SYS$SYSTEM:SYE

SYE VERSION X.X
_INPUT FILE:    [SYS:ERRORLOG:ERRLOG.OLD]  ?<CR>
_OUTPUT FILE:   [SYS$OUTPUT]               ?ERRLOG.DAT
_OPTIONS:       [ROLL-UP]                  ?R
_DEVICE NAME:   [(CR)]                     ?<CR>
_AFTER DATE:    [FIRST ENTRY]              ?<CR>
 BEFORE DATE:   [LAST ENTRY]               ?<CR>
$PRINT ERRLOG.DAT
```

RUNNING THE SYSTEM DUMP ANALYZER (SDA)

When the operating system crashes, the kernel routine writes the contents of the error log file, the processor registers, and physical memory to a contiguous file called SYSDUMP.DMP. With the help of the SDA commands, you can analyze and display parts of the formatted system dump file on a video display terminal. Or, you can create hardcopy listings.

Any user may run SDA by typing the DCL command:

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

When you issue this command, SDA will prompt for the name of the system dump file you want to examine.

```
Enter name of dump file >
```

To examine the most recent system dump, press RETURN at the prompt:

```
Enter name of dump file >
```

SDA will search the system directory [SYSEXE] (logical name SYS$SYSTEM) for the SYSDUMP.DMP file. To examine an older system dump, enter its file specification.

```
Enter name of dump file >DBO.[EYSEXE]SYSDUMP.OLD
```

SAVING THE SYSTEM DUMP FILE

When the kernel routine writes data into SYSDUMP.DMP, it destroys the previous contents of the file. Therefore, be sure to make a copy of the file under another name. Use the VMS copy command or the SDA copy command to save the file.

```
$ COPY SYSDUMP.DMP SAVEDUMP.DMP

      or

SDA>COPY SYS$SYSTEM:SAVEDUMP.DMP
```

SDA COMMANDS

Command	Function
<escape key>	Repeat last command
COPY	Copy dump file
DEFINE	Define symbols and their values
EVALUATE	Perform computations
EXAMINE	Examine memory locations
EXIT	Exit from display or utility
FORMAT	Format data blocks
READ	Copy object module symbols
SET OUTPUT	Set output to device or file specification
SET PROCESS	Set to the current process context
SHOW CRASH	Display crash information
SHOW DEVICE	Display I/O data structures
SHOW PAGE_TABLE	Display system page table
SHOW PFN_DATA	Display PFN data base
SHOW POOL	Display dynamic memory
SHOW PROCESS	Display specific process information
SHOW STACK	Display process/interrupt stacks
SHOW SUMMARY	Display process summary
SHOW SYMBOL	Display symbol table

Three help files within SDA will provide explanations.

- HELP<command-name> Briefly explains a command
- HELP SDA Briefly explains command format
- HELP Briefly explains the SDA utility

FILE TRANSFER UTILITY (FLX) HELP FILE

FLX Extensions

Qualifier	File Type
/IM (Image Mode)	TSK, OLB, SYS, SML, ULB, EXE
/FB (Formatted Binary)	OBJ, STB, BIN, LDA
/FA (Formatted ASCII)	All others
/CO (Contiguous)	For making a file copied to a disk contiguous.

Before entering the Files utility (FLX) you must mount the floppy:

```
$ MOUNT/For CSA1:
```

To enter FLX> running under VMS type:

```
$ MCR FLX  (Note: The $ is the prompt for VMS.)
```

The proper response to $MCR FLX should be:

```
FLX>
```

If the floppy has never been used and needs to be initialized to RT-11 format:

```
FLX>CS1:/RT/ZE
```

To copy from disk (Files-11 format (RS)) to floppy (RT-11 format (RT))

```
FLX>CS1:/RT=FILENAME.EXT/RS/IM or FB or FA
```

To copy from floppy to disk:

```
FLX>/RS=CS1:FILENAME.EXT/RT/IM or FB or FA
```

To copy from floppy to disk and make the file on disk contiguous:

```
FLX>/RS/CO=CS1:FILENAME.EXT/RT/IM or FB or FA
```

To delete one file on floppy:

```
FLX>CS1:FILENAME.EXT/RT/DE
```

To obtain a directory of the floppy:

```
FLX>CS1:/RT/LI
```

To copy SRØBOO.CMD from floppy to disk:

```
FLX>/RS=CS1:SRØBOO.CMD/RT/FA
```

To copy CONSOL.SYS from floppy to disk:

```
FLX>/RS=CS1:CONSOL.SYS/RT/IM
```

FILE TRANSFER UTILITY (FLX) HELP FILE (CONT)

To copy DRØBOO.CMD from disk to floppy:

```
FLX>CS1:/RT=DRØBOO.CMD/RS/FA
```

To copy ESSAA.EXE from disk to floppy:

```
FLX>CS1:/RT=ESSAA.EXE/RS/IM
```

To delete a file from the floppy:

```
FLX>CS1:DEFBOO.CMD/RT/DE
```

To copy ESSAA.EXE from floppy to disk contiguous:

```
FLX>RS/CO=CS1:ESSAA.EXE/RT/IM
```

To make a file bootable, you must use the utility RTB. Before using RTB, enter the following assignment using DCL:

```
$ ASSIGN SYS$DISK: TRØ
```

Then invoke RTB by:

```
$ MCR RTB
```

The correct response should be:

```
RTB>
```

Then perform the following:

```
RTB>CS1:=FILENAME.EXT
```

Example:

```
RTB>CS1:=CONSOL.SYS
```

To exit either FLX> or RTB>, type CTRL/Z (^Z).

TERMINAL FUNCTION KEYS

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

EDT VERSION 2 VT100 KEYPAD

<table>
<tr><td>GOLD</td><td>HELP</td><td>FNDNXT
FIND</td><td>DEL L
UND L</td></tr>
<tr><td>PAGE
COMMAND</td><td>SECT
FILL</td><td>APPEND
REPLACE</td><td>DEL W
UND W</td></tr>
<tr><td>ADVANCE
BOTTOM</td><td>BACKUP
TOP</td><td>CUT
PASTE</td><td>DEL C
UND C</td></tr>
<tr><td>WORD
CHNGCASE</td><td>EOL
DEL EOL</td><td>CHAR
SPECINS</td><td rowspan="2">ENTER
SUBS</td></tr>
<tr><td colspan="2">LINE
OPEN LINE</td><td>SELECT
RESET</td></tr>
</table>

BACKSPACE	GO TO BEGINNING OF LINE
DELETE	DELETE CHARACTER
LINEFEED	DELETE TO START OF WORD
CTRL/A	COMPUTE TAB LEVEL
CTRL/D	DECREASE TAB LEVEL
CTRL/E	INCREASE TAB LEVEL
CTRL/K	DEFINE KEY
CTRL/T	ADJUST TABS
CTRL/U	DELETE TO START OF LINE
CTRL/W	REFRESH SCREEN
CTRL/Z	RETURN TO LINE MODE

TK-8976

EDT VERSION 2 VT52 KEYPAD

GOLD	HELP	DEL L UND L	UP REPLACE
PAGE COMMAND	FNDNXT FIND	DEL W UND W	DOWN SECT
ADVANCE BOTTOM	BACKUP TOP	DEL C UND C	RIGHT SPECINS
WORD CHNGCASE	EOL DEL EOL	CUT PASTE	LEFT APPEND
LINE OPEN LINE		SELECT RESET	ENTER SUBS

BACKSPACE	GO TO BEGINNING OF LINE
DELETE	DELETE CHARACTER
LINEFEED	DELETE TO START OF WORD
CTRL/A	COMPUTE TAB LEVEL
CTRL/D	DECREASE TAB LEVEL
CTRL/E	INCREASE TAB LEVEL
CTRL/F	FILL TEXT
CTRL/K	DEFINE KEY
CTRL/T	ADJUST TABS
CTRL/U	DELETE TO START OF LINE
CTRL/W	REFRESH SCREEN
CTRL/Z	RETURN TO LINE MODE

TK-8977

CHAPTER 6
DIAGNOSTICS

Diagnostic	Level	Title	Floppy
BLDRIVER	N/A	VMS I/O Driver for ESDIC	ESZ4E
DNDRIVER	N/A	VAX/VMS DN11 Device Driver	ESZ2C
DR750	N/A	DR750 Functional Microcode	ESZ7B
DR780	N/A	DR780 Functional Microcode	ESZ7B
DVDRIVER	N/A	VAX/VMS DV11 Device Driver	ESZ3C
EMDRIVER	N/A	VAX/VMS ML11 Driver	EVZ5E
EC0AB	N/A	VAX-11/750 Microcode Documentation	None
EC0AD	N/A	VAX-11/750 Boot ROM Documentation	None
ECCAA	3	VAX-11/750 RH750 Diagnostic	MFG1
ECCBA	3	VAX-11/750 UBI/DW750 Diagnostic	None
ECDFA	3	DR750 Micromachine	None
ECDFB	3	DR750 Silomachine	None
ECDFK	3	DR750 APT Microcode Loader	None
ECKAA	4	VAX-11/750 Microdiagnostic Monitor	None
ECKAB	4	VAX-11/750 Microdiagnostic DPM	None
ECKAC	4	VAX-11/750 Microdiagnostic MIC	None
ECKAF	4	VAX KC750 Microdiagnostic	None
ECKAL	4	VAX-11/750 Cache/TB Diagnostic	MFG1
ECKAM	3	VAX-11/750 Memory Diagnostic	MFG2
ECKAX	3	VAX-11/750 Cluster Exerciser	MFG2
ECSAA	N/A	VAX-11/750 Diagnostic Supervisor	MFG2
ECVMB	N/A	VAX-11/750 Boot Block Documentation	None
ENKAA	5	VAX-11/730 Micro-Monitor	MFG3
ENKAX	3	VAX-11/730 Cluster Exerciser	MFG4
ENKBA	5	VAX-11/730 8085 Test, Part 1	MFG3
ENKBB	5	VAX-11/730 8085 Test, Part 2	MFG3
ENKBC	5	VAX-11/730 8085 Test, Part 3	MFG3
ENKBD	5	VAX-11/730 8085 Test, Part 4	MFG3
ENKBE	5	VAX-11/730 8085 Test, Part 5	MFG3
ENKBF	5	VAX-11/730 WCS March Test	MFG3
ENKBG	5	VAX-11/730 APT/RD Usart Modem Loopback Test	MFG3

VAX DIAGNOSTIC PROGRAM CODES (CONT)

Diagnostic	Level	Title	Floppy
ESCAA	3	VAX MBA (RH780) Diagnostic	ESZAE
ESCBA	3	VAX UBA (DW780) Diagnostic	ESZAE
ESCCA	3	MA780 Multiport Exerciser	ESZ9B
ESCCB	3	MA780 Multiport Memory Exerciser	ESZ3E
ESCDA	3	IEC11-CA Functional Diagnostic	ESZ6B
ESCGA	3	CI780 Repair Level, Part 1	ESZ8E
ESCGB	3	CI780 Repair Level, Part 2	ESZ8E
ESCGC	3	CI780 Repair Level, Part 3	ESZ8E
ESCGD	3	CI780 Repair Level, Part 4	ESZ8E
ESDCA	2R	VAX Synchronous Link Diagnostic	ESZ3B,ESZ9A
ESDEA	2	VAX Communication Multiplexer Diagnostic	ESZ3C
ESDF	N/A	DR780 Microcode	ESZ7B
ESDFA	3	DR780 High Speed SBI Interface, Part A	ESZ6B
ESDFB	3	DR780 High Speed SBI Interface, Part B	ESZ6B
ESDFC	3	DR780 High Speed SBI Interface, Part C	ESZ6B
ESDFD	3	DR780 Functional Self-Test	ESZ7B
ESDFE	3	DR780 Functional Drive-to-Drive Test	ESZ7B
ESDFF	2	DR780 Functional Self-Test	ESZ8B
ESDFG	2	DR780 Functional Drive-to-Drive Test	ESZ8B
ESDFH	2	DR780 Start-Up	ESZ8B
ESDGA	3	DQS11-A/B Repair Level	ESZ5C
ESDIC	2	KMS11-BL/BM Functional Level	ESZ4E
ESDID	3	VAX KMS11-BL/BM Repair Level	ESZ4E
ESDPA	2R	VAX PCL11B Exerciser	ESZ7A
ESDPB	3	VAX PCL11B Repair Diagnostic	ESZ7A
ESDRA	3	VAX DR11B Repair Diagnostic	ESZ5C
ESDXB	3	Communications IOP (M8206) Repair Level	ESZ3D
ESDXC	2	Communications IOP Functional Diagnostic	ESZ3D
ESKAA	4	VAX-11/780 Local Console (KA780) Program	ESZAB,ESZAC, ESZAD,ESZAG
ESKAB	4	VAX-11/780 (KA780) Microdiagnostic Monitor	ESZAC,ESZAD, ESZBC,ESZBD, ESZFC,ESZFD, ESZAG
ESKAC	4	VAX-11/780 (KA780) Hardcore Monitor	ESZAC,ESZBC, ESZFC

Diagnostic	Level	Title	Floppy
ESKAD	4	VAX-11/780 (KA780) Hardcore Test Stream	ESZAC,ESZBC, ESZFC
ESKAE	4	VAX-11/780 (KA780) Micro Test Monitor	ESZAC,ESZBC, ESZAD,ESZBD, ESZAG,ESZFC, ESZFD
ESKAF	4	VAX-11/780 (KA780) Microdiagnostic Command Parser	ESZAC,ESZBC, ESZAD,ESZBD, ESZAG,ESZFC, ESZFD
ESKAG	4	VAX-11/780 (KA780) Microdiagnostic, Directory Search Routine	ESZAC,ESZBC, ESZAD,ESZBD, ESZAG,ESZFC, ESZFD
ESKAH	4	VAX-11/780 (KA780) Microdiagnostic, Volume I	ESZAC,ESZBC, ESZFC
ESKAJ	4	VAX-11/780 (KA780) Microdiagnostic, Volume II	ESZAD,ESZBD, ESZFD
ESKAK	4	VAX-11/780 (KA780) Microdiagnostic Fail Chain Monitor	ESZAC,ESZBC, ESZAD,ESZBD, ESZAG,ESZFC, ESZFD
ESKAL	4	VAX-11/780 (KA780) Fail Chain, Volume I	ESZAC,ESZBC, ESZFC
ESKAM	4	VAX-11/780 (KA780) Fail Chain, Volume II	ESZAD,ESZBD, ESZFD
ESKAN	4	VAX MA780 Microdiagnostic Go Chain	ESZAG
ESKAP	4	VAX MA780 Microdiagnostic Fail Chain	ESZAG
ESKBB	4	VAX-11/780 Remote Console (KA780 Program)	ESZBB,ESZBC, ESZBD
ESKCC	4	VAX-11/780 European Remote Console (KA780) Program	ESZFB,ESZFC, ESZFD
ESSAA	N/A	VAX-11/780 Diagnostic Supervisor	ESZAE,ESZFC
ESTGA	3	VS70 Graphics Interface Diagnostic	ESZ2D

Diagnostic	Level	Title	Floppy
ESTGB	3	VS70 Vector Generator Diagnostic	ESZ2D
ESTGC	3	VS70 Memory and Microprocess Diagnostic	ESZ2D
ESTGD	3	VS70 IPCU-Tablet Diagnostic	ESZ2D
ESTGE	2R	VS70 Memory and Display Test	ESZ2D
ESTGF	2R	VS70 IPCU-Tablet Diagnostic	ESZ2D
ESUBA	N/A	VAX Diagnostic Disk Build Command File	ESZDD
ESUBB	N/A	VAX Diagnostic Update Command File	ESZDD
ESOAA	4	WCS Microcode Documentation	ESZAB,ESZAD, ESZBB,ESZBD, ESZFB,ESZFD
ESOAB	4	PCS Microcode Documentation	ESZAB,ESZBB, ESZFB
ESOAC	4	FPA PCS Microcode Documentation	N/A
ESOAD	4	11/780 Memory Bootstrap	N/A
EVAAA	2R	VAX Line Printer Diagnostic	ESZ3B
EVABA	2R	VAX CR11 CR Diagnostic Test	ESZ3B
EVALA	2	LPA11-K System	ESZ2B,ESZ5B
EVALB	3	AA11-K D/A Converter Repair	ESZ5B
EVALC	3	AA11-K A/D Converter Repair	ESZ4B
EVALD	3	DR11-K DIGITAL I/O Interface	ESZ4B
EVALE	3	KW11 Dual Programmable Clock	ESZ4B
EVALF	3	LPA11-K I/O Bus Test	ESZ4B
EVCFA	2R	DEC Dataway Exerciser	ESZ9A
EVDAA	3	VAX DZ11 8 Line Synchronous Multiplexer	ESZ7C
EVDAB	3	DZ32 Repair Level Diagnostic	ESZ2E
EVDAC	2R	DZ32 Level 2R Functional	ESZ2E
EVDBA	3	VAX M8201/2 Repair Level Diagnostic	ESZ9A
EVDBB	3	VAX DMC11 Exerciser Program	ESZ9A
EVDDA	3	DR11-C Repair Level	ESZ2C
EVDDB	2	DR11-C Functional Level	ESZ2C
EVDEA	2	VAX Communication Multiplexer Diagnostic	ESZ3C
EVDEB	3	DV11 Basic R/W Microprocessor Instruction Exerciser	ESZ3C
EVDEC	3	DV11 Static Line Card Test	ESZ4C
EVDED	3	DV11 Free Running ROM Test, Part 1	ESZ4C

Diagnostic	Level	Title	Floppy
EVDEE	3	DV11 Free Running ROM Test, Part 2	ESZ4C
EVDEF	3	DV11 Modem Control and Label Test	ESZ9C
EVDEG	3	DV11 Asynchronous Line Card Test	ESZ9C
EVDFD	3	DR780/750 Self-Test Functional	ESZ7B
EVDFE	3	DR780 Drive-to-Drive Diagnostic	ESZ7B
EVDFF	2R	DR780/750 User Mode Self-Test	ESZ8B
EVDFG	2R	DR780/750 User Mode Drive-to-Drive Diagnostic	ESZ8B
EVDFI	3	DF11-W Repair Level Diagnostic	ESZ2C
EVDFJ	2	DF11-W Level 2 Diagnostic	ESZ2C
EVDHA	3	KMC11-B Repair Diagnostic	ESZ3D
EVDHB	2	VAX Level 2 KMC11-B Diagnostic	ESZ3D
EVDIA	3	VAX KMS11-BD Repair Level	ESZ9D
EVDIB	2	VAX KMS11-BD Level 2 Diagnostic	ESZ9D
EVDJA	3	DM11-BA Modem Control Multiplexer	ESZ9D
EVDKA	3	DMS11-DA Repair Diagnostic	ESZ9D
EVDMA	3	M8203 Repair Level	ESZ7C
EVDMB	2R	VAX DMP11 Functional Diagnostic	ESZ7C
EVDMC	2	VAX DMC11/DMR11 Data Communications Link	ESZ9A
EVDMD	2R	VAX DMP11 Data Communications Link Test	ESZ1F
EVDNA	2	VAX/DN11 ACU Interface Diagnostic	ESZ2C
EVDNB	3	DN11 Auto Call Unit Interface Test	ESZ2C
EVDRB	2R	DR11W On-Line Diagnostic	ESZ5C
EVDRE	3	DR11W Repair Level Diagnostic	ESZ5C
EVDSA	3	DRS11/DSS11 Repair Level Diagnostic	ESZ5D
EVDUP	3	DUP11 Repair Level Diagnostic, Part 1	ESZ3A
EVDUQ	3	DUP11 Repair Level Diagnostic, Part 2	ESZ3A
EVDXA	3	VAX Communications IOP Repair Diagnostic	ESZ7C,ESZ5B, ESZ9A
EVKAA	4	VAX Hardcore Instruction Test	ESZAF
EVKAB	2	VAX Architectural Instruction	ESZAF
EVKAC	2	VAX Floating-Point Instruction	ESZAH
EVKAD	2	VAX Compatibility Mode Instruction	ESZAH
EVKAM	2R	VAX Memory User Mode Test	
EVKAE	3	VAX Privileged Architecture Instruction	ESZAH
EVKAM	2R	VAX Memory User Mode Test	ESZAH

Diagnostic	Level	Title	Floppy
EVMAA	2	VAX TMØ3/TE16/TU45-77 Tape	ESZ1B,ESZ6C
EVMAB	3	VAX TMØ3-TE16/TU45-77 Drive Functional Timer	ESZ2A
EVMAC	3	VAX TMØ3/TE16/TU45/TU77	ESZ2A
EVMAD	3	VAX TS11 Subsystem Repair	ESZ6C
EVMAE	3	TM78/TU78 Control Logic	ESZ7D
EVQBL	N/A	Standalone Driver for ESDIC	ESZ4E
EVQDB	N/A	Loadable Driver for RPØ4/5/6	ESZ1A
EVQDD	N/A	Loadable Driver for TU58	ESZ1A
EVQDL	N/A	Loadable Driver for RLØ2	ESZ3B,ESZ1A
EVQDM	N/A	Loadable Driver for RKØ6/7	ESZ1A,ESZ3B
EVQDN	N/A	Loadable Driver for DN11	ESZ2C
EVQDR	N/A	Loadable Driver for RMØX/8Ø/RP7	ESZ1A,ESZ3B, ESZ8C
EVQDV	N/A	Standalone Driver for DV11	ESZ3C
EVQDY	N/A	Loadable Driver for RXØ2	ESZ1A
EVQEM	2	Loadable Driver for ML11	ESZ1A,ESZ3B
EVQIB	N/A	Loadable Driver for IEC11-B	ESZ6D
EVQIE	N/A	Loadable Driver for IEC11-A	ESZ6D
EVQKM	N/A	Loadable Driver for KMC11-B	ESZ3D,ESZ9D
EVQSB	N/A	DB-11A Device Driver (SBDRV)	ESZ9A
EVQTF	N/A	Loadable Driver for TU78	ESZ1B,ESZ1C
EVQTM	N/A	Loadable Driver for TMØ3-TE16/TU77	ESZ1B
EVQTS	N/A	Loadable Driver for TS11	ESZ6C
EVQUE	N/A	VAX UBE Driver	ESZ4D
EVQVS	N/A	Loadable Driver for VS11	ESZ1E
EVQXA	N/A	Loadable Driver for DR11-W/DF11	ESZ2C,ESZ5C
EVQXC	N/A	Loadable Driver for COM IOP	ESZ3D
EVQXD	N/A	VAX DMP11 Diagnostic Driver	ESZ1F
EVQXF	N/A	Loadable Driver for DR78Ø	ESZ8B
EVQXM	N/A	Loadable Driver for DMR11/DMC11	ESZ7C,ESZ9A
EVRAA	2	VAX RP/RK/RM/RX/TU58 Reliability	ESZ1A,ESZ3B
EVRAC	2	VAX Disk Formatter	ESZ1A,ESZ3B
EVRBA	3	VAX RPØX Functional Diagnostic	ESZ6A
EVRCA	3	VAX DCL/RPØ4,5,6 Repair	ESZ6A

Diagnostic	Level	Title	Floppy
EVRDA	3	VAX RMØ3/RMØ5/RMØ8 Diskless	ESZ1C,ESX8C
EVRDB	3	VAX RMØ3/RMØ5 Function Test	ESZ1C
EVREA	3	VAX RK611 Diagnostic, Part A	ESZ4A
EVREB	3	VAX RK611 Diagnostic, Part B	ESZ4A
EVREC	3	VAX RK611 Diagnostic, Part C	ESZ5A
EVRED	3	VAX RK611 Diagnostic, Part D	ESZ5A
EVREE	3	VAX RK611 Diagnostic, Part E	ESZ5A
EVREF	3	VAX RK611-RKØ6/Ø7 Drive Functional Test, Part 1	ESZ8A
EVREG	3	VAX RK611-RKØ6/Ø7 Drive Functional Test, Part 2	ESZ8A
EVRFA	3	RLØ2 Subsystem Functional Diagnostic	ESZ1D
EVRGA	3	RM8Ø Formatter	ESZ8C
EVRGB	3	RM8Ø Functional Diagnostic	ESZ8C
EVRHA	3	RPØ7 Front End Diagnostic	ESZ8D
EVRHB	3	RPØ7 Functional Diagnostic	ESZ8D
EVRHC	3	RPØ7 Dual Port Control Diagnostic	ESZ8D
EVRIA	3	VAX RXØ2 Subsystem Repair	ESZ1D
EVRJA	3	VAX ML11 Diagnostic	ESZ5E
EVRJB	3	ML-11 Exerciser	ESZ5E
EVRLA	3	VAX UDA5Ø Disk Subsystem Diagnostic	ESZ2F
EVRLB	2	VAX UDA5Ø Disk Formatter	ESZ2F
EVSBA	3	VAX Autosizer	ESZ4D
EVTAA	2R	VAX Terminal Diagnostic	ESZ2B
EVTBA	2R	VAX Terminal Exerciser	ESZ2B
EVTCA	2	VSV11 Display Level 2 Diagnostic	ESZ1E
EVTCB	3	VAX VS11 Level 3 Diagnostic	ESZ1E
EVXBA	2R	VAX Bus Interaction Program	ESZ4D
EVXBB	2R	VAX System Diagnostic	ESZ4D
GSDRIVER	N/A	VAX/VMS VS7Ø Device Driver	ESZ2D
KMDRIVER	N/A	VAX/VMS KMC11-B Device Driver	ESZ9D
RCDRIVER	N/A	VAX/VMS DR11C Device Driver	ESZ2C
TTDRIVER	N/A	VAX DZ32 Diagnostic Terminal Driver	ESZ2E
UETML1100	N/A	VAX/VMS ML11 ICP for UETP	ESZ5E
VSDRIVER	N/A	VAX/VMS VS11 Device Driver	ESZ1E
XCDRIVER	N/A	VAX/VMS COM IOP Device Driver	ESZ3D
XDDRIVER	N/A	VAX/VMS DMP11 Device Driver	ESZ7C

PRIVILEGES AND QUOTAS NEEDED TO RUN DIAGNOSTICS ON LINE

Privilege	Quotas
GPRNAM	CLI:DCL
ALLSPOOL	ASTLM: 1000
DETACH	DIOLM: 1000
DIAGNOSE	WSDE FAULT: 256
LOG_IO	PCRCLM: 100
GROUP	PIOLM: 1000
PRMCEB	FILLM: 100
PRMMBX	WSQUOTA: 512
PSWAPM	PRI
TMPMBX	TGELM: 1000
WORLD	PGFLQUOTA: 40
PHY_IO	

SET LOAD COMMAND

SET LOAD <device>:[directory]<CR>

The SET LOAD command establishes the storage device from which the supervisor will load diagnostic programs. The default load device is the device from which the supervisor was booted. Use SET LOAD when you wish to load diagnostic programs from a different device. Use the SET LOAD command in combination with the LOAD command or the RUN command.

```
DS> SET LOAD DMA0:[SYSMAINT]
DS> LOAD ESDXA

DS> SET LOAD DMA0:[SYSMAINT]
DS> RUN ESDXA
```

SET LOAD Command Example

NOTE

The directory name, and the square brackets around it, are necessary in the SET LOAD command.

SHOW LOAD COMMAND

SHOW LOAD<CR>

The SHOW LOAD command causes the supervisor to display the storage device from which diagnostic programs are to be loaded when the LOAD command is given.

```
DS> SHOW LOAD
DMA0:[SYSMAINT]
DS>
```

SHOW LOAD Command Example

LOAD COMMAND

LOAD <file-spec><CR>

This command loads the specified file into main memory from the default load device. The default file extension is .EXE. The storage device from which the program is loaded is the device established on the previous SET LOAD command. Note that you need supply only the five-letter code that identifies each diagnostic program for the command line argument <file-space>.

```
LOAD ESTAA          ! Load the local terminal
                    ! diagnostic program.
```

LOAD Command Example

ATTACH COMMAND

ATTACH <UUT-type><link-name><generic-device-name>...<CR>

The operator must use several ATTACH commands, before starting a diagnostic program, to define each unit under test (UUT), and the devices that link it to the SBI, for the supervisor. If you are testing several units at once, repeat the ATTACH command for each device. Every unit under test is uniquely defined by a hardware designation and a link.

The first parameter, <UUT-type>, is the hardware designation of the unit under test. For example, RH78Ø, TMØ3, TE16, and DZ11 are hardware designations.

The second parameter, <link-name>, is the name of the piece of hardware that links the unit under test, in most cases through intermediate links, to the main system bus. For example, an RH78Ø is linked to the SBI; a TU45 is linked to an MTa; and a DZ11 is linked to a DWn. You must attach each piece of hardware (with the exception of the SBI) before you can use it as a link in an ATTACH command.

The third parameter is the generic device name, which identifies to the supervisor the particular unit to be tested. Use the form GGan for the device name. GG is a two-character generic device name (alphabetic); a is an alphabetic character, specifying the device controller; n is a decimal number in the range of Ø-255, specifying the number of the unit with respect to the controller.

Use n or a only if it is applicable to the device. You must supply additional information for some types of hardware to enable the diagnostic program to address the device. For example, you must supply TR and BR numbers for an RH78Ø, the controller number for a TMØ3, and the CSR, vector, and BR for a UNIBUS device. If you do not include additional information, but the information is necessary, the supervisor will prompt you for it.

In the generic name:

a is a letter from A to Z
n is a decimal number in the range Ø-255
?? is a generic device name that may be any two letters

```
DS> ATTACH DW780 SBI DW0 3 4          ! Attach the DW780.
DS> ATTACH DZ11 DW0 TTA               ! Attach the DZ11 TTA.
CSR? 760120                           ! The supervisor prompts
VECTOR?  320                          ! for information not
BR?  4                                ! supplied in the command
                                      ! line.
DS>
```

ATTACH Command Example

SELECT COMMAND

```
SELECT <generic-device-name>[:],-<CR>
[<generic-device-name>[:]...]!ALL<CR>
```

The operator must select each unit to be tested with the SELECT command, after attaching it. For each unit, supply the appropriate generic device name. SELECT adds the specified device to the list of units to be tested. The command takes effect the next time the diagnostic program is started.

```
DS> SELECT   TTA:
DS>
```

SELECT Command Example

DESELECT COMMAND

```
DESELECT <device>[:][,<device>[:]...]!ALL<CR>
```

Use the DESELECT command to remove the name of one or more devices from the list of units to be tested.

```
DS> DESELECT   TTA:
DS> DESELECT   ALL
DS>
```

DESELECT Command Example

SHOW DEVICE COMMAND

```
SHOW DEVICE <device>[:][,<device>[:]...]<CR>
```

The SHOW DEVICE command causes the supervisor to display the characteristics of the specified devices on the operator's terminal. If you omit the device name, the supervisor will list the characteristics of all attached devices (see the following example).

SHOW SELECTED COMMAND

SHOW SELECTED<CR>

The SHOW SELECTED command causes the display of information in the same format as the SHOW DEVICE command. However, the information is shown only for the devices that have been previously selected.

```
DS> SHOW DEVICE
_DW0     DW780                  60006000  TR=3. BR=4. NUMBER=0.
_DMA     RK611      _DW0        6013FF20  CSR=00000777440(O) VECTOR=00000000210(O) BR=5.
_DMA0    RK07       _DMA        00000000
_TTA     DZ11       _DW0        6013E050  CSR=00000760120(O) VECTOR=00000000320(O) BR=4.

DS> SHOW SELECTED

DS> SELECT TTA:
DS> SHOW SELECTED
 TTA     DZ11       _DW0        6013E050  CSR=00000760120(O) VECTOR=00000000320(O) BR=4.
DS> DESELECT TTA:
DS> SHOW SELECTED
DS>
```

SHOW DEVICE and SHOW SELECTED Commands Example

START COMMAND

START [/SECTION:<section-name>]-<CR>
[/TEST:<first>[:<last>!/SUBTEST:<num>]]-<CR>
[/PASSES:<count>]<CR>

The START command causes the diagnostic supervisor to pass control to the initialize routine in the diagnostic program in memory, thus beginning program execution.

Each diagnostic program is organized in discrete tests. The tests are grouped in sections, according to their functions, execution times, and whether or not there is a need for operator interaction.

If the START command is given without switches, the program will run the tests in the default section. In other words, the initial setting for SECTION is DEFAULT. The supervisor calls only those tests that have been designed by the diagnostic engineer to run in the default section. Default section tests do not require operator intervention. When a section is selected in conjunction with the START command, only the tests that 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 number. If the SUBTEST switch is used in conjunction with the PASSES switch, the operator is provided with a loop-on-subtest capability. In this case, only the subtest named in the command line is executed, once looping begins.

If the TEST switch is not specified, all tests within the named section of the program are executed. In other words, the default for TEST is TEST a through TEST n, where TEST n is the highest numbered test in the section. If only the first argument is specified with the TEST switch, the last argument is assumed to default to be the highest numbered test within the selected section of the program.

Tests are run only if they are included in the section named. If the PASSES switch is not used, the default value is 1. Test and pass numbers are decimal. The minimum value for passes is 1. The maximum value is Ø, which means infinity in this context.

For example:

```
DS> START                              ! Start execution of the
                                       ! diagnostic program in memory.

DS> START/SEC:MANUAL                   ! Start execution of the manual
                                       ! section of the program.

DS> START/SEC:MANUAL/TEST:32:33        ! Run tests 32 and 33 if they are
                                       ! in the manual section.  Some
                                       ! tests may not be executed unless
                                       ! the section is specified.

DS> START/TEST:6:12                    ! Run tests 6, 7, 8, 9, 1Ø, 11, 12.

DS> START/TEST:9/SUBTEST:5             ! Run test 9, subtests 1, 2, 3,
                                       ! 4, 5.

DS> START/TEST:9                       ! Run tests 9 through n, where n
                                       ! is the last test in the default
                                       ! section.

DS> START/PASS:3                       ! Run 3 passes of the default
                                       ! section.

DS> START/TEST:9/SUBTEST:5/PASS:Ø      ! Execute test 9, subtests 1, 2,
                                       ! 3, 4, and then loop on subtest
                                       ! 5 indefinitely.
```

START Command Example

RUN COMMAND

```
RUN <file-spec>[/SECTION:<section name>]-<CR>
    [/TEST:<first>[:<last>!/SUBTEST:<num>]]-<CR>
    [/PASSES:<count>]<CR>
```

RUN is equivalent to a LOAD and START command sequence. The RUN command switches are identical to those in the START command.

For example:

```
DS> RUN ESTAA                          ! Load and run the local
                                       ! terminal diagnostic.

DS> RUN ESTAA/SEC:MANUAL               ! Load the local terminal
                                       ! diagnostic and run the
                                       ! manual section.

DS> RUN ESTAA/SEC:MANUAL/TEST:32:33    ! Load the local terminal
                                       ! diagnostic and run tests
                                       ! 32 and 33 in the manual
                                       ! section.

DS> RUN ESTAA/TEST:6:12                ! Load the local terminal
                                       ! diagnostic and run tests
                                       ! 6, 7, 8, 9, 10, 11, 12.

DS> RUN ESTAA/TEST:9/SUBTEST:5         ! Load the local terminal
                                       ! diagnostic and run test 9,
                                       ! subtests 1, 2, 3, 4, 5.

DS> RUN ESTAA/TEST:9                   ! Load the local terminal
                                       ! diagnostic and run tests 9
                                       ! through n, where n is the
                                       ! last test in the default
                                       ! section.

DS> RUN ESTAA/PASS:3                   ! Load the local terminal
                                       ! diagnostic and run three
                                       ! passes.

DS> RUN ESTAA/TEST:9/SUBTEST:5/PASS:0  ! Load the local terminal
                                       ! diagnostic, execute test 9,
                                       ! subtests 1, 2, 3, 4, and then
                                       ! loop on test 9, subtest 5,
                                       ! indefinitely.
```

RUN Command Example

SUMMARY COMMAND

SUMMARY<CR>

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. Type CTRL/C first to return control to the command line interpreter (CLI). Then type SUMMARY to obtain a statistical report on the program. CONTINUE may be typed at this point, if the operator wishes to resume program execution.

CTRL/C

Normally CTRL/C returns control from a diagnostic program to the command line interpreter in the diagnostic supervisor. The supervisor then enters a command wait state and displays the DS> prompt on the operator's terminal. The operator may then issue any valid command. CTRL/C is the only diagnostic supervisor command that may be issued while a program is running. When a diagnostic program is running in conversation mode, CTRL/C returns control to a command interpreter within the program for the conversation mode.

CONTINUE COMMAND

CONTINUE<CR>

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

The following example shows how CTRL/C, SUMMARY, and CONTINUE can be used together to obtain statistics on the program run and to then resume execution.

```
                    ...Program is running...

^C                                  ! Operator types CTRL/C.
DS> SUMMARY                         ! Supervisor prompt.
                                    ! Operator requests
                                    ! statistical report.

                          Statistical
                            Report

DS> CONTINUE                        ! Supervisor prompt.
                                    ! Operator requests
                                    ! resumption of program.

                    ...Program is running...
```

Example of Use of CTRL/C, SUMMARY, and CONTINUE Commands

ABORT COMMAND

ABORT<CR>

This command passes control to the program's cleanup code and then returns control to the supervisor, which enters a command wait state and displays the supervisor prompt, DS>. At this point the operator may issue any command except CONTINUE. The following example shows how the ABORT command can be used together with CTRL/C and SUMMARY.

```
                    ...Program is running...

^C                                  ! Operator types CTRL/C.
DS> SUMMARY                         ! Supervisor prompt.
                                    ! Operator requests
                                    ! statistical report.

                         Statistical
                           Report

DS> ABORT                           ! Supervisor prompt.
                                    ! Operator requests program
                                    ! cleanup and termination.

DS>                                 ! Supervisor prompt.
```

Example of Use of CTRL/C, SUMMARY, and ABORT Commands

SET FLAGS COMMAND

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

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 and 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. When set, this flag causes the program to enter a predetermined scope loop on a test or subtest that detects a failure. Set the IE1 flag if you want to inhibit error messages. Looping will continue until the operator returns control to the supervisor by using the CTRL/C command. The operator may then continue, clear the flag and continue, or abort the program.
VERIFY	SET VERIFY FOR .COM FILES.

BELL	Bell on error. When set, this flag causes the supervisor to send 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 that 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 information message (first three lines) 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. When set, this flag indicates to the program that the operator wants a quick verify mode of operation. The interpretation of this flag is program dependent.
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.
OPERATOR [DEFAULT]	Operator present. When set, this flag indicates to the supervisor that operator interaction is possible. When cleared, the supervisor takes appropriate actions to ensure that the test session continues without an operator.
PROMPT [DEFAULT]	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><CR>

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<CR>
```

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

SHOW FLAGS COMMAND

```
SHOW FLAGS<CR>
```

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

The following example shows how the SET FLAGS, CLEAR FLAGS, and SHOW FLAGS commands can be coordinated.

```
DS> SET FLAGS TRACE              ! Set the TRACE flag.
DS> CLEAR FLAGS QUICK            ! Clear the QUICK flag.
DS> SHOW FLAGS
CONTROL FLAGS SET: PROMPT, OPER, TRACE
CONTROL FLAGS CLEAR: QUICK, IES, IE3, IE2, IE1, BELL, LOOP, HALT

DS>
```

Example of the Use of the FLAG CONTROL Commands

SET EVENT FLAGS COMMAND

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

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.

Event flags are status posting bits maintained by VMS and the supervisor. Diagnostic programs can use event flags to perform a variety of signaling functions, including communication with the operator.

CLEAR EVENT FLAGS COMMAND

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

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]<CR>

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

The following example shows how the SET EVENT FLAGS, CLEAR EVENT FLAGS, and SHOW EVENT FLAGS commands can be coordinated.

```
DS> SET EVENT FLAGS 1, 9, 15
DS> CLEAR EVENT FLAGS 2, 6
DS> SHOW EVENT FLAGS
EVENT FLAGS SET: 15, 9, 1
DS>
```

Example of EVENT FLAGS CONTROL Commands

SET BASE COMMAND

SET BASE <address><CR>

This 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 PC numbers provided in the listings can be used directly in referencing locations in the program sections.

For example:

```
DS> SET BASE E00        ! Set the base
                        ! address to the
                        ! beginning of the psect of
                        ! the routine under
                        ! examination.
DS>
```

SET BASE Command Example

NOTE

Virtual address = physical address (normally) when memory management is turned off.

SET BREAKPOINT COMMAND

SET BREAKPOINT <address><CR>

This command causes control to pass to the supervisor when the program counter points to the <address> previously specified by this command. A maximum of 15 simultaneous breakpoints can be set within the diagnostic program.

For example:

```
DS> SET BREAKPOINT    30        ! Set a breakpoint
                                ! at an offset of
                                ! 30 from the
                                ! base address.
```

SET BREAKPOINT Command Example

CLEAR BREAKPOINT COMMAND

CLEAR BREAKPOINT <address>!ALL<CR>

This command clears the previously set breakpoint at the memory location specified by <address>. If no breakpoint existed at the specified address, no error message is given. An optional argument of ALL clears all previously defined breakpoints.

For example:

```
DS> CLEAR BREAKPOINT    30      ! Clear the breakpoint
                                ! at the location which
                                ! is offset 30 from
                                ! the base address.
DS>
```

CLEAR BREAKPOINT Command Example

SHOW BREAKPOINTS COMMAND

SHOW BREAKPOINTS<CR>

This command displays all currently defined breakpoints.

For example:

```
DS> SHOW BREAKPOINTS            ! Display breakpoints
                                ! currently set.
CURRENT BREAKPOINTS:
        00000E30(X)
DS>
```

SHOW BREAKPOINTS Command Example

SET DEFAULT COMMAND

```
SET DEFAULT <argument-list><CR>
```

This command causes setting of default qualifiers for the examine and deposit commands. The <argument-list> argument consists of data length default and/or radix default qualifiers. If both qualifiers are present, they are separated by a comma. If only one default qualifier is specified, the other one is not affected. Initial defaults are HEX and LONG. Default qualifiers are:

```
Data Length: Byte, Word, Long
Radix: Hexadecimal, Decimal, Octal
```

For example:

```
DS> SET DEFAULT BYTE, DECIMAL ! Set the default data
                              ! length qualifier as
                              ! byte and the default
                              ! radix qualifier as
                              ! decimal.
DS>
```

SET DEFAULT Command Example

EXAMINE COMMAND

(ONLINE = VIRTUAL, OFFLINE = PHYSICAL)

```
EXAMINE [<qualifiers>][<address>]<CR>
```

The EXAMINE command displays the contents of memory in the format described by the qualifiers. If no qualifiers are specified, the default qualifiers set by a previous default command are used. The applicable qualifiers are described in the following table.

Examine Command Qualifier Descriptions

Qualifier	Description
/B	Address points to a byte
/W	Address points to a word
/L	Address points to a longword
/H	Display in hexadecimal radix
/D	Display in decimal radix
/O	Display in octal radix
/A	Display in ASCII bytes

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

For example:

```
DS> EXAMINE 30              ! Display the contents
                            ! of the longword which
                            ! is offset 30 from
                            ! the base address of E00.
00000E30:  D0513D01
DS>
```

EXAMINE Command Example

DEPOSIT COMMAND

DEPOSIT [<qualifiers>]<address><data><CR>

This command accepts data and writes it into the memory location specified by <address> in the format described by the qualifiers. If no qualifiers are specified, the default qualifiers are used. The applicable qualifiers are identical to those of the EXAMINE command described in the previous table.

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

For example:

```
DS> DEPOSIT/W/H 30 0001     ! Deposit 0001 (hex)
                            ! in the word
                            ! offset 30 from
                            ! the base address.
00000E30:  0001
DS>
```

DEPOSIT Command Example

NEXT COMMAND

```
NEXT [number-of-instructions]<CR>
```

This command causes the supervisor to execute one macro-instruction. If you specify a number (decimal) after NEXT, the supervisor will execute that number of macroinstructions. The supervisor displays the PC of the next instruction and the contents of the next four bytes, after execution of each instruction.

Use this command to step through an area of a program where you suspect a problem. Do not use the NEXT command unless you have stopped the program at a breakpoint.

For example:

```
DS> NEXT                          ! Execute the next instruction.
00000E31:  D0513D01
DS>
```

NEXT Command Example

SHOW SECTIONS (LOAD DIAG FIRST)
SHOW STATUS (SIMILAR TO SUMMARY)
HELP DEVICE DDMA (ATTACH CONFIG HELP)

DEVICE NAMING CONVENTIONS

UUT-Type	Link	Generic	Parameters	Typical
AA11K	DWn	??an	CSR VCT BR	770460 350 5
AD11K	DWn	??an	CSR VCT BR	770400 xxx 6
CR11	DWn	CRa	CSR VCT BR	777160 230 4
DL11	DWn	??a	CSR VCT BR	
DMC11	DWn	XMa	CSR VCT BR	760050 xxx 5
DMP11	DWn	XDan	CSR VCT BR	
DMR11	DWn	XMan	CSR VCT BR	
DR11B	DWn	??a	CSR VCT BR	772410 124
DR11K	DWn	??a	CSR VCT BR	767770 xxx 4
DR11W	DWn	??a	CSR VCT BR	
DR780	SBI	XFn	TR BR	
DUP11	DWn	XJa	CSR VCT BR	
DV11	DWn	XVa	CSR VCT BR	775000
DW750	CMI	DWn	BR	
DW780	SBI	DWn	TR BR	3 4 (#1 UBA)
DW780	SBI	DWn	TR BR	4 4 (#2 UBA)
DZ11	DWn	TTa	CSR VCT BR EIA/20MIL	760100 xxx 5 EIA
KA750	CMI	KAn	G H TOY WCS ACC	NO NO YES 0 0
KA780	SBI	KAn	G H WCS ACC	NO NO 0 0
KMC11	DWn	XMan	CSR VCT BR	
KW11K	DWn	??a	CSR VCT BR	770404 xxx 6
LA34	TTa	TTan		
LA36	TTa	TTan		
LA38	TTa	TTan		
LA120	TTa	TTan		
LA180	LPa	LPan		
LP05	LPa	LPan		
LP06	LPa	LPan		
LP11	DWn	LPa	CSR VCT BR	777514 200 4
LP14	LPa	LPan		
LP25	LPa	LPan		
LPA11K	DWn	LAan	CSR VCT BR	770460 350 5
MA780	SBI	MAn	TR BR MPM PORT	
MBE	RHn	MBn	DRIVE #	
MS750	CMI	MSn	BR	
MS780	SBI	MSn	TR	
PCL11	DWn	??a	CSR VCT BR	764200 170 x
RH750	CMI	RHn	BR	5
RH780	SBI	RHn	TR BR	8 5 (RH0)
RH780	SBI	RHn	TR BR	9 5 (RH1)
RK06	DMa	DMan		
RK07	DMa	DMan		
RK611	DWn	DMa	CSR VCT BR	777440 210 5
RL01	DLa	DLan		
RL02	DLa	DLan		
RL11	DWn	DLa	CSR VCT BR	774400 160 5
RM03	RHn	DRan		
RM05	RHn	DRan		
RM80	RHn	DRan		
RP04	RHn	DRan		
RP05	RHn	DBan		
RP06	RHn	DBan		
RP07	RHn	DBan		
RX02	DYa	DYan		
RX211	DWn	DYa	CSR VCT BR	777170 264 5
TE16	MTa	MTan		
TM03	RHn	MTa	DRIVE #	

DEVICE NAMING CONVENTIONS (CONT)

UUT-Type	Link	Generic	Parameters	Typical
			DRIVE #	
TM78	RHn	MFa		
TS11	DWn	MSan	CSR VCT BR	772520 224 5
TU45	MTa	MTan		
TU58	DWn	DDan	CSR VCT BR	776500 xxx x
TU77	MTa	MTan		
TU78	MFa	MFan		
UBE	DWn	UBan	CSR VCT BR	
VT50	TTa	TTan		
VT52	TTa	TTan		
VT55	TTa	TTan		
VT100	TTa	TTan		

Note: The typical column is only a partial list because of the great number of possible configurations.

a = alpha character.
n = numeric character.

DS> HELP DEVICE <UUT>

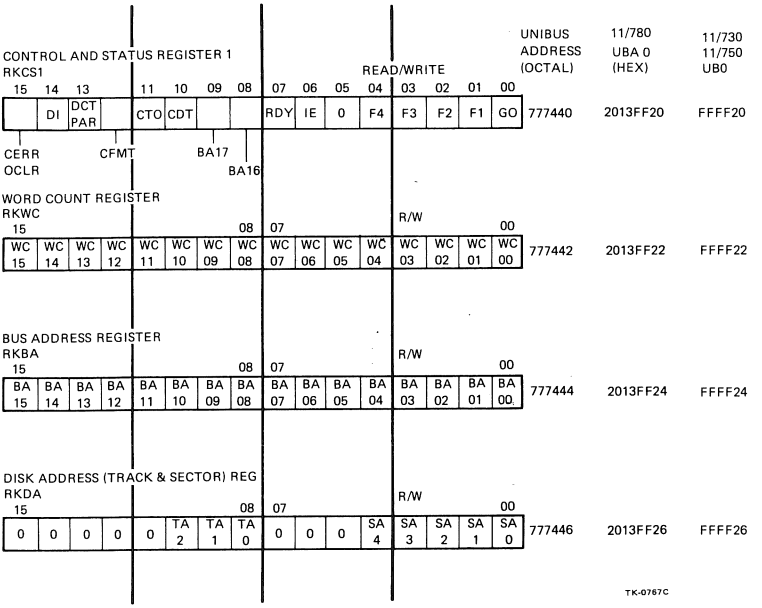
RA60 RA80 RA81 LINK: DUa NAME: DUan

UDA50 LINK: DWa NAME: DUa CSR: 772150 (DEF)
VEC: FLOATING (154) BR: 5

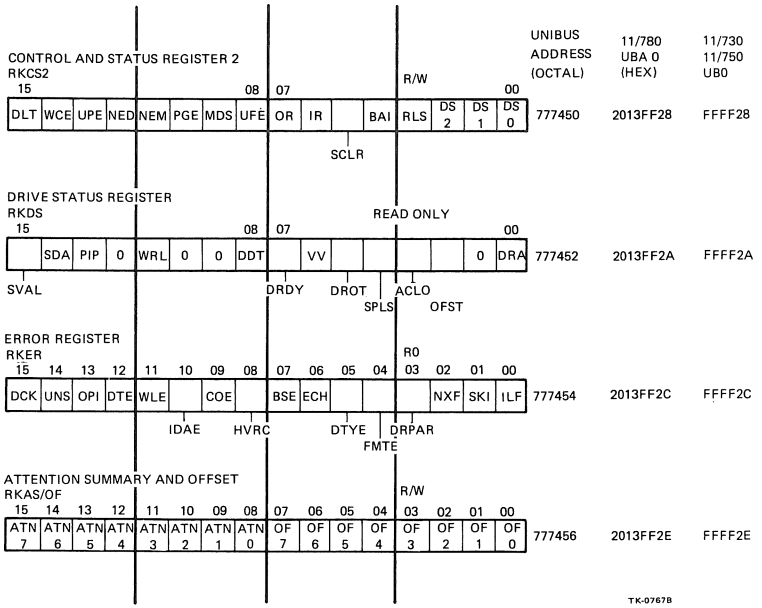
DMF32

CHAPTER 7
I/O OPTIONS

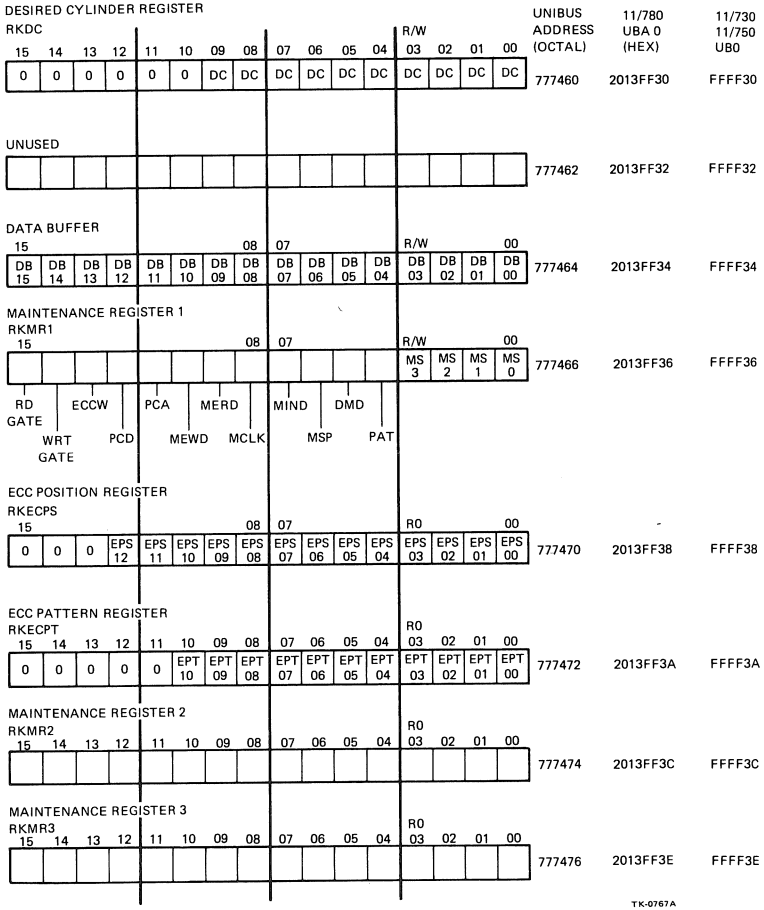
RK611/RK711 REGISTERS BIT CONFIGURATION



RK611/RK711 REGISTERS BIT CONFIGURATION (CONT)



RK611/RK711 REGISTERS BIT CONFIGURATION (CONT)



DZ11 REGISTER BIT CONFIGURATION

	BYTES: HIGH								LOW								UNIBUS ADDRESS (OCTAL)	11/780 UBA 0 (HEX)	11/730 11/750 UB0
	MSB 15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	LSB 00			
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**	FFE040
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**	FFE042
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**	FFF042
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**	FFE044
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**	FFE046
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**	FFE046

*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

11/750 1st RH750 F28400
2nd RH750 F2A400
3rd RH750 F2C400

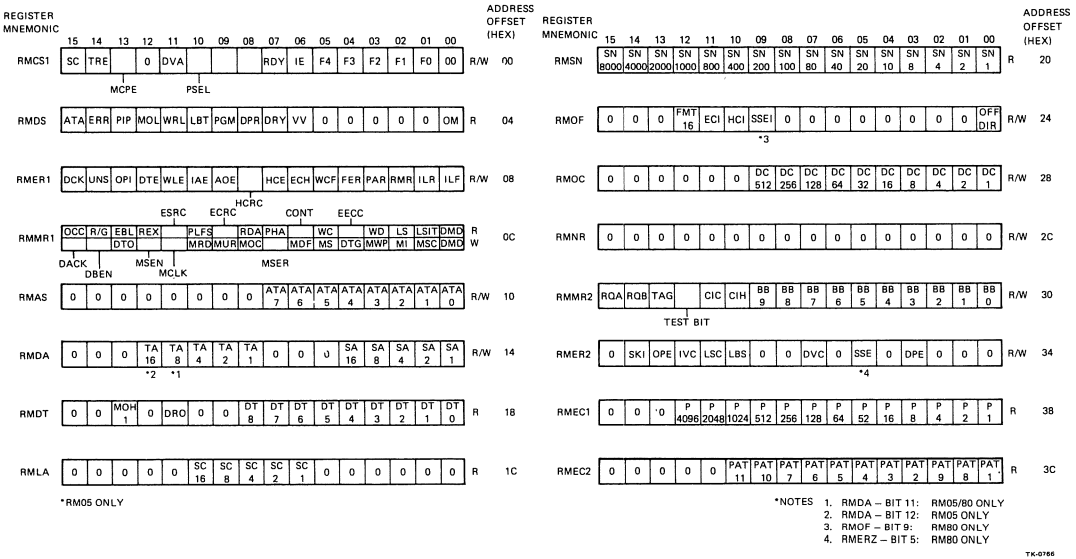
11/780 1st RH780 20010400 (TR8)
2nd RH780 20012400 (TR9)
3rd RH780 20014400 (TR10)
4th RH780 20016400 (TR11)

Register Number		Drive Type			Drive Number							
Hex	Octal	RP (Disk)	RM (Disk)	TE (Tape)	0	1	2	3	4	5	6	7
0	0	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

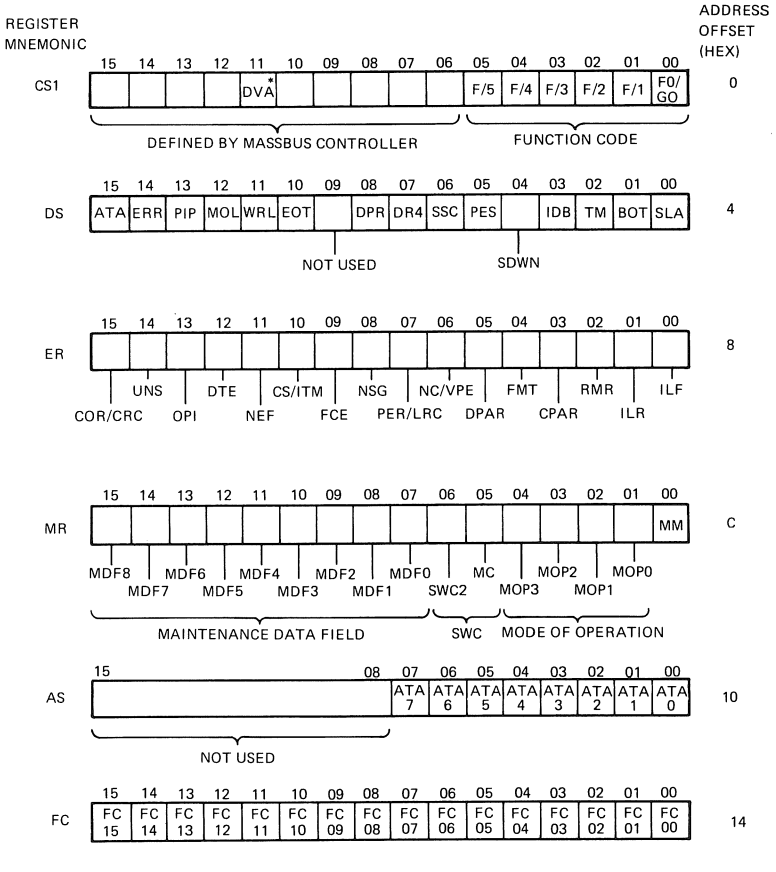
REGISTER MNEMONIC	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	ADDRESS OFFSET (HEX)
RPCS1	DVA										F4	F3	F2	F1	F0	GO	00
RPDA	0	0	0	TA 16	TA 8	TA 4	TA 2	TA 1	0	0	0	SA 16	SA 8	SA 4	SA 2	SA 1	14
RPDS	ATA	ERR	PIP	MOL	WRL	LST	PGM	DPR	DRY	VV	0	0	0	0	0	0	04
RPER1	DCK	UNS	OPI	DTE	WLE	IAE	AOE	HCRC	HCE	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	DRO	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

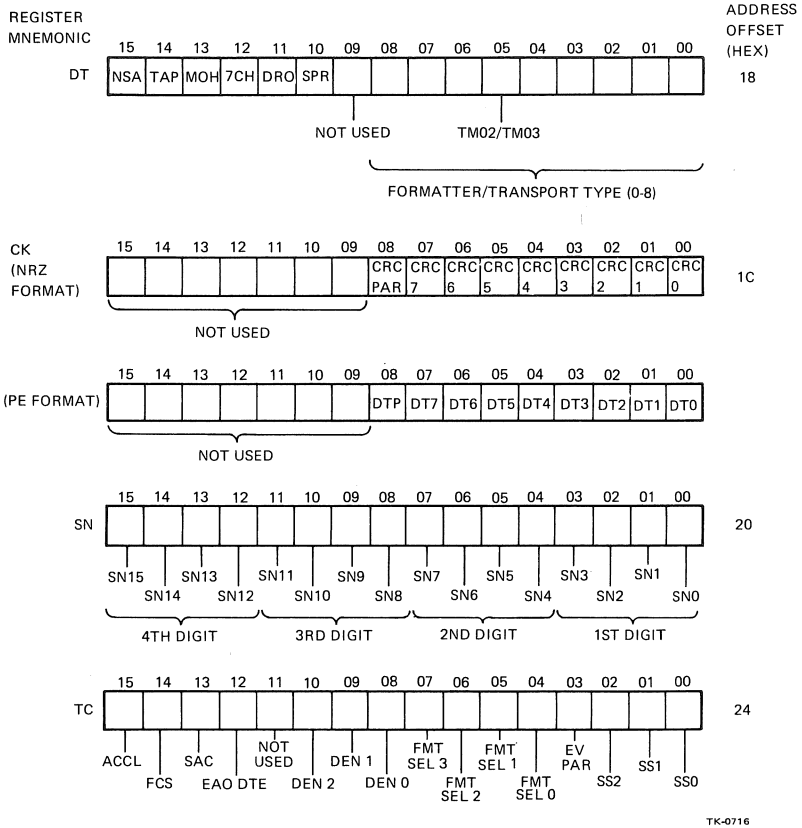

REGISTER MNEMONIC
ADDRESS OFFSET (HEX)
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
RMCS1 SC TRE 0 DVA RDY IE F4 F3 F2 F1 F0 00 R/W 00
MCPE
PSEL
RMDS ATA ERR PIP MOL WRL LBT PGM DPR DRY VV 0 0 0 0 0 OM R 04
RMER1 DCK UNS OPI DTE WLE IAE AOE HCE ECH WCF FER PAR RMR ILR ILF R/W 08
HCRC
ESRC ECRC CONT 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 TA 16 TA 8 TA 4 TA 2 TA 1 0 0 0 SA 16 SA 8 SA 4 SA 2 SA 1 R/W 14
*2 *1
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
*RM05 ONLY
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 SSEI 0 0 0 0 0 0 0 OFF DIR R/W 24
*3
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 SSE 0 DPE 0 0 0 R/W 34
*4
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
*NOTES 1. RMDA – BIT 11: RM05/80 ONLY
2. RMDA – BIT 12: RM05 ONLY
3. RMOF – BIT 9: RM80 ONLY
4. RMERZ – BIT 5: RM80 ONLY
TK-0765

MASSBUS Registers	Bits																Address Offset (Hex)
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RPCS1	0	0	0	0	DVA	0	0	0	0	0	F4	F3	F2	F1	F0	G0	R/W	00
RPDS	ATA	ERR	PIP	MOL	WRL	LBT	PGM	DPR	DRY	VV=1	0	0	0	ILV	EWN	OM	R	04
RPER1	DCK	UNS	OPI	DTE	WLE	IAE	AOE	HCRC	HCE	ECH	WCF	FER	PAR	RMR	ILR	ILF	R	08
RPMR1	DMD	DIAG 64	DIAG 32	DIAG 16	DIAG 8	DIAG 4	DIAG 2	DIAG 1	P	A	R	A	M	E	T	E R	R/W	0C
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	R/W	10
RPDA	0	0	TA 32	TA 16	TA 8	TA 4	TA 2	TA 1	0	SA 64	SA 32	SA 16	SA 8	SA 4	SA 2	SA 1	R/W	14
RPDT	0	0	1	0	DRQ	0	0	0	0	0	1	0	0	0	DT 1	0	R	18
RPLA	0	0	0	SC 64	SC 32	SC 16	SC 8	SC 4	SC 2	SC 1	0	0	0	0	0	0	R	1C
RPSN	8K	4K	2K	1K	800	400	200	100	80	40	20	10	08	04	02	01	R	20
RPOF	CMD	MTD	0	FMT 16	ECI	HCI	0	0	0	0	0	0	0	0	0	0	R/W	24
RPDC	0	0	0	0	0	0	512	256	128	64	32	16	8	4	2	1	R/W	28
RPCC	0	0	0	0	0	0	512	256	128	64	32	16	8	4	2	1	R	2C
RPER2	PGE	CPE	CPU	RWU 3	RWU 2	RWU 1	WOR	WRY UNS	E	R	R	-	C	O	D	E	R	30
RPER3	BSE	SKI	DSE	WSF	CLF	LBC	LCE	PHF	DVC	IXU	DCU	SDF	DPE	SBE	SCF	RTO	R	34
RPEC1	0	0	0	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	R	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	R	3C

TM03 REGISTER CONTENTS



TM03 REGISTER CONTENTS (CONT)



RH780	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	TYPE
BASE+0	0				DVA	0					DT FUNCTION CODE					GO	1
4	DT FAILURE CODE						0	DPR	0		DT INTERRUPT CODE						1
8	SER	FORMAT			SKIP COUNT				RECORD COUNT						CMD ADR		1
C	PRINT FLGS	ERROR MSG NR							DIAG TEST NR								1*
10	0								ATTENTION BIT								1
14	BYTE COUNT																1
18	NSA	TAP	0		DUAL MB	0	WCS	DRIVE TYPE (101)									1
1C	RDY	PRES	ONL	REW	PE	BOT	EOT	FPT	AVAIL	SHR	MAINT	DSE	0	0	0	0	1
20	BCD SN 3				BCD SN 2				BCD SN 1				BCD SN 0				1
24	AUX PRINT NR				DATA PATTERN NR				LOOP	$\overline{OV}$	0	COMP	DIAG REQ				1*
28	EXPECTED DIAG DATA								ACTUAL DIAG DATA								1*
2C	NDT FAILURE CODE						ATTN ADR		0	0	NDT INTERRUPT CODE						1
30	COMMAND COUNT 0								0	0	NDT FUNCTION CODE 0					GO	1
34	COMMAND COUNT 1								0	0	NDT FUNCTION CODE 1					GO	1
38	COMMAND COUNT 2								0	0	NDT FUNCTION CODE 2					GO	1
3C	COMMAND COUNT 3								0	0	NDT FUNCTION CODE 3					GO	1
40	INTERNAL ADDRESS																2
44	TM RDY	TM CLR	MC PE	ILR	CPE	EV PAR	HLDA	HOLD	INTERNAL DATA								2
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

TYPE
1. COMMON ADDRESS SPACE
2. TM78 HARDWARE CONTROL REGISTERS

* DIAGNOSTIC USE ONLY

TM78 REGISTER CONTENTS (CONT)

Byte	Description
1	Command code being executed on last error
2	Interrupt code from last error
3	Failure code from last error
4	Hardware register Ø; read path write fail bits
5	Hardware register 1; read path diagnostic bits
6	Hardware register 2; read path status
7	Hardware register 3; read path command loop
8	Hardware register 2Ø; AMTIE
9	Hardware register 21; RC DONE
1Ø	Hardware register 22; illegal 5-to-4
11	Hardware register 23; Mark 2
12	Hardware register 24; End Mark
13	Hardware register 25; RC PAR bits
14	Hardware register 26; postamble det
15	Hardware register 27; data
16	Hardware register 3Ø; CRC
17	Hardware register 31; corrected data
18	Hardware register 32; ECC status
19	Hardware register 4Ø; channel Ø TIE bus
2Ø	Hardware register 41; channel 1 TIE bus
21	Hardware register 42; channel 2 TIE bus
22	Hardware register 43; channel 3 TIE bus
23	Hardware register 44; channel 4 TIE bus
24	Hardware register 45; channel 5 TIE bus
25	Hardware register 46; channel 6 TIE bus
26	Hardware register 47; channel 7 TIE bus
27	Hardware register 5Ø; channel P TIE bus
28	Hardware register 6Ø; TIE bus
29	Hardware register 1Ø4; AMTIE

TM78 REGISTER CONTENTS (CONT)

Byte	Description
30	Hardware register 110; PORT status
31	Hardware register 114; read data
32	Hardware register 240; CAS status
33	Hardware register 241; CBUS status
34	Hardware register 300; DBUS status
35	Hardware register 320; WMC status
36	Hardware register 321; TU select 0
37	Hardware register 322; TU select 1
38	Hardware register 323; write data
39	Hardware register 324; byte counter <7:0>
40	Hardware register 324; byte counter <15:8>
41	hardware register 325; PAD counter <7:0>
42	Hardware register 325; PAD counter <15:8>
43	Hardware register 326; ECODE counter <7:0>
44	Hardware register 326; ECODE counter <15:8>
45	Hardware register 330; DDR/MBD A
46	Hardware register 331; DDR/MBD B
47	Hardware register 332; WMC errors
48	Hardware register 340; interrupt status
49	MIA register 0; TU78 status
50	MIA register 1; MIA status A
51	MIA register 2; MIA status B
52	MIA register 3; serial NR A
53	MIA register 4; serial NR B
54	MIA register 5; TU diagnostics
55	Retry counter (RETCNT). This byte is the count of retry interrupt requests given for the tape unit. When this count is zero, the tape unit is not in a retry sequence.

Byte	Description
56	Retry control bits (RETCNT+1). This byte is used by the microcode to control error recovery. It is meaningful only when the retry counter (byte 55) is not zero. Bit 5 - Set when initial command moved in the reverse direction. Bit 6 - Set when initial command was a read. Bit 7 - Set when last retry requested was in opposite direction of initial command.
57	TU software status (TUx). This byte contains information about the tape drive. Bit Ø - Set when a DATA SECURITY ERASE command is in progress. Bit 1 - Set when a REWIND command is in progress. Bit 2 - Set when tape unit exists and power is on. Bit 3 - Set when a NONDATA TRANSFER command issued from a MASSBUS port is in progress. Bit 4 - Set when tape was last moved in the reverse direction. Bit 5 - Set when last tape operation involved writing on tape. Bit 6 - Set when last record seen was a tape mark. Bit 7 - Set when last MASSBUS command came from port B.
58	Transfer control word (XFRCTL). This byte contains control information used by DATA TRANSFER commands. Bits Ø-2 - Write clock select Bits 3-5 - Read clock select Bits 6 - PLO bypass Bits 7 - Low read threshold
59	Retry suppress and format control (XRETRY). This byte contains the contents of the left half of the MASSBUS register, which contains the retry suppress bit, format, and skip count.
6Ø	Keypad enable flag (ENAON). This byte is not zero when the keypad is enabled.

CHAPTER 8

MISCELLANEOUS

CONVERSION TABLES

HEX ADDER

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

CONVERSION TABLES (CONT)

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

CONVERSION TABLES (CONT)

HEX/DECIMAL CONVERSION

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

CONVERSION TABLES (CONT)

HEX/OCTAL CONVERSION

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

CONVERSION TABLES (CONT)

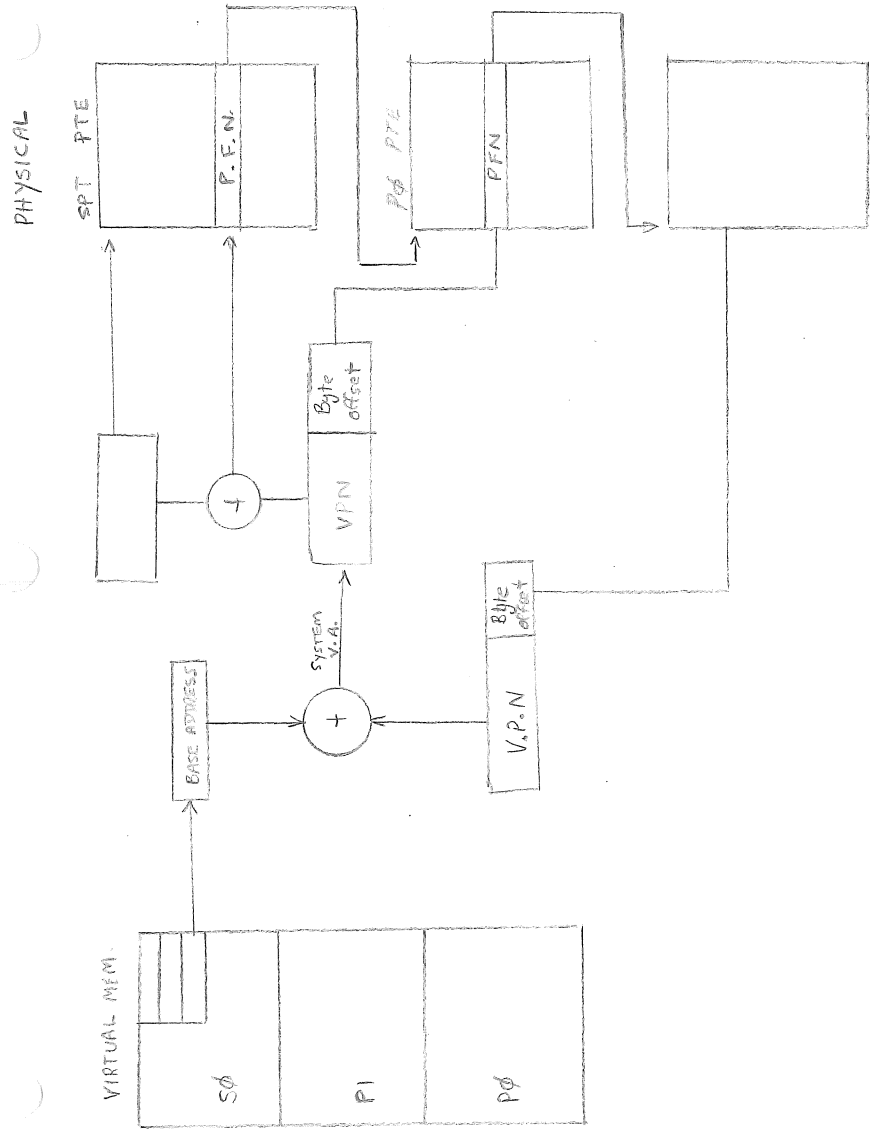
OCTAL/DECIMAL CONVERSION

	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				

CONVERSION TABLES (CONT)

HEX/ASCII CONVERSION

HEX Code	ASCII Char	HEX Code	ASCII Char	HEX Code	ASCII Char	HEX Code	ASCII Char
ØØ	NUL	2Ø	SP	4Ø	@	6Ø	`
Ø1	SOH	21	!	41	A	61	a
Ø2	STX	22	"	42	B	62	b
Ø3	ETX	23	#	43	C	63	c
Ø4	EOT	24	$	44	D	64	d
Ø5	ENQ	25	%	45	E	65	e
Ø6	ACK	26	&	46	F	66	f
Ø7	BEL	27	'	47	G	67	g
Ø8	BS	28	(	48	H	68	h
Ø9	HT	29	)	49	I	69	i
ØA	LF	2A	*	4A	J	6A	j
ØB	VT	2B	+	4B	K	6B	k
ØC	FF	2C	'	4C	L	6C	l
ØD	CR	2D	-	4D	M	6D	m
ØE	SO	2E	.	4E	N	6E	n
ØF	SI	2F	/	4F	O	6F	o
1Ø	DLE	3Ø	Ø	5Ø	P	7Ø	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


PHYSICAL
SPT PTE
P.F.N.
P0 PTE
PFN
Byte offset
VPN
+
System V.A.
BASE ADDRESS
+
V.P.N
Byte offset
VIRTUAL MEM
S0
P1
P0

NOTES

NOTES

NOTES

NOTES

VAX MAINTENANCE HANDBOOK: VAX SYSTEMS

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 *Documentation Products Directory*, 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
Accessories and Supplies Group
P.O. Box CS2008
Nashua, New Hampshire 03061

Attention: Documentation Products
Telephone: 1-800-258-1710

Order No. EK-VAXV1-HB

Fold Here

Do Not Tear – Fold Here and Staple

digital

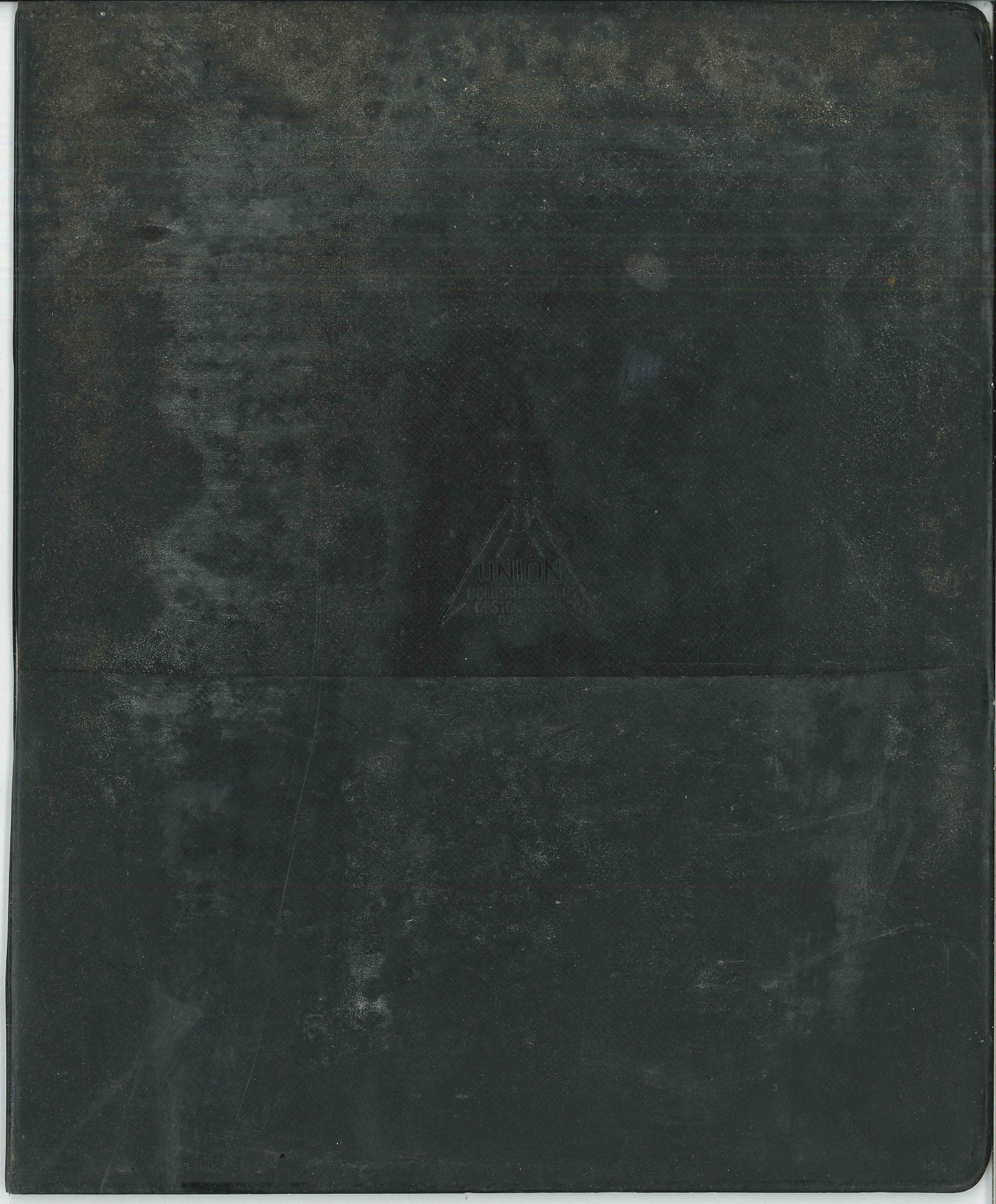
No Postage Necessary if Mailed in the United States

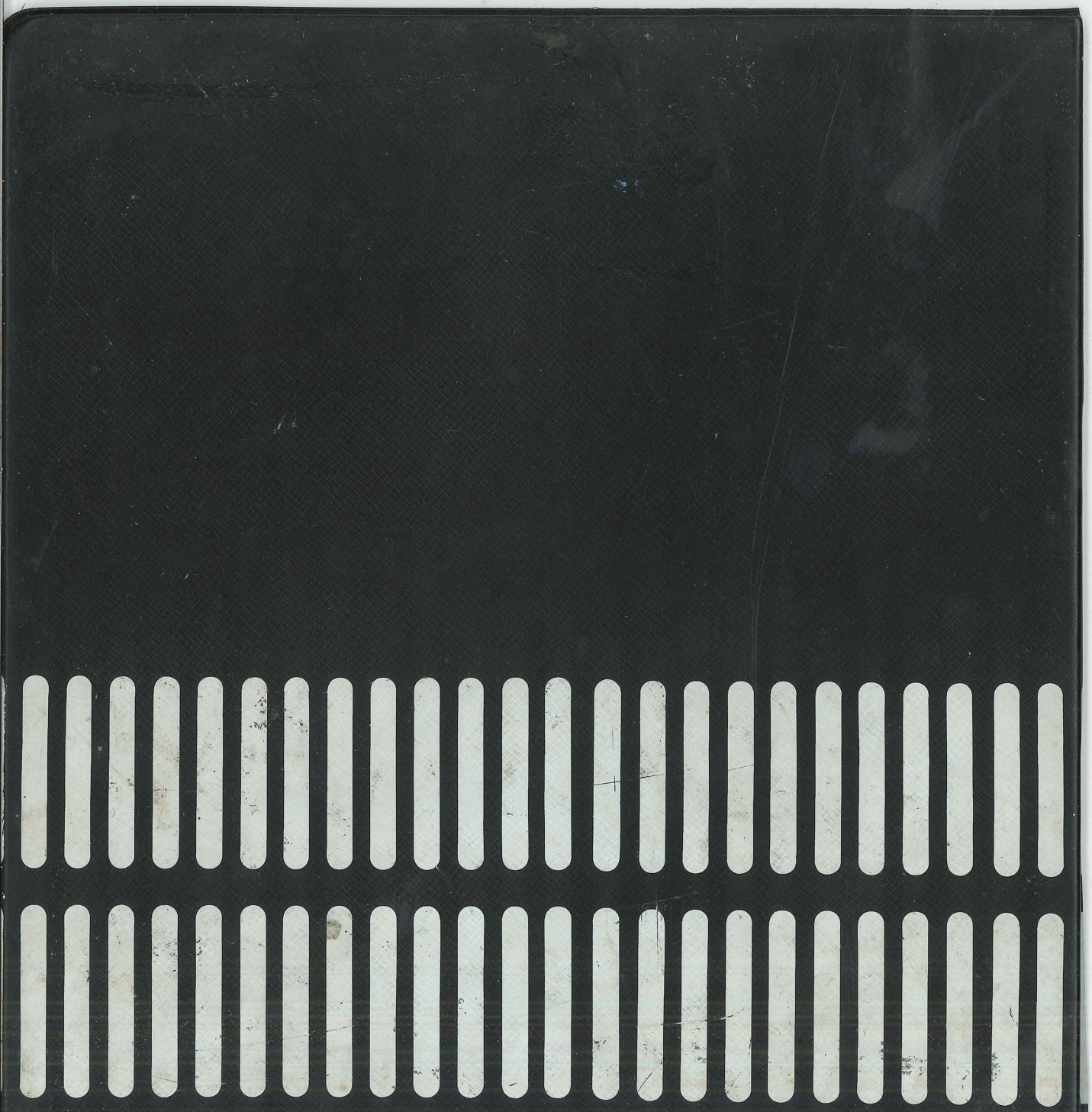
BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MA.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services/Quality Assurance
12 Crosby Drive, BU/E08
Bedford, MA 01730


UNION



Digital Equipment Corporation • Bedford, MA 01730