IDENTIFICATION

PRODUCT CODE. ZZ-ESOAA-124.0

PRODUCT TITLE: 11780 WRITE CONTROL STORE

PRODUCT DATE: DECEMBER 1981

DEPARTMENT: CURRENT PRODUCTS ENGINEERING

; Table of Contents

;
; Table of Contents

;
; Table of Contents

;
; Table of Contents

:
: Table of Contents

```
                                        ;1      .TOC    "REV.MIC"
                                        ;2      .TOC    "Revision 1(124).0"
                                        ;3      ;       P. R. Guilbault
                                        ;4
;5      .NOBIN
;6      .TOC    "       Comprehensive Revision History"
;7
;8      ;       The revision history that follows indicates which modules have changed and includes a SUMMARY of those changes.
;9      ; Individual revision histories should be consulted for descriptions of the changes. The location of each revision history
;10     ; is available from the table of contents.
;11
;12     ;1(124) CALCRC  Created by merging CALL2 with CRC
;13     ;       DECMAL  Fix (ADD,SUB)P(4,6) don't always set  V bit problem.
;14     ;               Fix (ADD,SUB)P(4,6) sign problem.
;15     ;               Fix CVTPT V bit problem.
;16     ;               Fix CVTTP interrupt problem.
;17     ;               Fix ASHP setting PSL<v> on negative shift count
;18     ;               Fix ASHP doesn't always set PSL<v> when positive shift is greater than destination length.
;19     ;       CHAR    Optimize MOVC5 fill char routine with quad writes.
;20     ;       EDIT    Restarting with FPD set and junk in R0 on reserved operands.
;21     ;               FPD set on reserved operand fault
;22     ;               Fix FPD unpack problem.
;23     ;       FLOAT   Fix POLYF arguement or partial product equal to zero problem.
;24     ;               Add floating faults for warm POLYF/D
;25     ;               Fix POLY backup up general regs on interrupt after FPD set
;26     ;               Fix warm POLYF/D rounding problem with small negative numbers
;27     ;       FPA     Fix POLYD rounding problem.
;28     ;               Convert hot floating traps to hot floating faults.
;29     ;               Fix POLY backup up general regs on interrupt after FPD set
;30     ;       GANDH   First inclusion of G & H microcode
;31     ;       IE      Fix compatibility mode odd address abort bad PC.
;32     ;       MOVPR   Add "MTPR 63" new [illegible] check.
;33     ;               Fix MTPR SCBB checks bits<27:2> for MBZ, should check bits<30:31>
;34     ;       PATCH   Change two patches to fix ACB[illegible]G) and G converts.
;35     ;               Modify patch 025 to fix (ADD,[illegible])P4 v bit.
;36     ;               Fix ACBD with neg ADDEND, branch on GTR instead of GEQ. No FPLA
;37     ;       PROBE   Add patch no. 091 to fix PROBEx C bit problem.
;38     ;       REI     Fix compatibility mode odd address abort bad PC.
;39     ;               Change files to get rid of absolute jumps :
;40     ;               ARITH,ASPC,CALL2,CNSL,CRC,DECMAL,FIELD,FLOAT,FORKS,FPA,IE,MMFW,MOVPR,PATCH,QUEUE
;41     ;1(123) QUEUE   Fix release 2nd interlock on error routine.
;42     ;               Add several missing quad alignment checks. Also wrong data being used in some places.
;43     ;               Fix interlocked queue's do not correctly restore PC on memory mng fault.
;44     ;1(122) FLOAT   Delete FLOATW.MIC and put in FLOAT
;45     ;       QUEUE   Delete Q1.MIC and put in QUEUE
;46     ;       DECMAL  Fix MULP X*Y produces different results than Y*X
;47     ;       PATCH   Fix MULP X*Y produces different results than Y*X
;48     ;               Remove PLA trap PCS040B to WCS117E. Not required add never implemented.
;49     ;               Remove PLA trap PCS040F to WCS117F. Not required add never implemented.
;50     ;               Remove PLA trap PCS03CF to WCS1150 from PATCH.MIC file. Not required add never implemented.
;51     ;               Remove PLA trap PCS0284 to WCS1151 from PATCH.MIC file. Not required add never implemented.
;52     ;1(121) Special revision for SAO that included G(but not H) floating without faults.
;53     ;1(120) Delete SPLIT expression, and merge PCS and WCS assemblies.
;54     ;       Reduce the no. of source code files by merging related code into 1 file
;55     ;       For revision history prior to this history, see DOC.MIC past history.
```

```
;56     ;       Start of history
```

```
;57      ;         P1W120  P1W121  P1W122  P1W123  P1W124  P1W125  P1W126  P1W127  P1W128  P1W129
;58      ;DOC      00                              01
;59
;60      ;DEFIN    00                              01
;61
;62      ;MACRO    00              01              02
;63
;64      ;PATCH    00              01      02      03
;65
;66      ;FORKS    00                              01
;67
;68      ;ARITH    00                              01
;69
;70      ;INDEX    00                              01
;71
;72      ;FLOAT    00              01              02
;73
;74      ;INIT2    00
;75
;76      ;ASPC     00                              01
;77
;78      ;FIELD    00                              01
;79
;80      ;CHAR     00                              01
;81
;82      ;EDIT     00                              01
;83
;84      ;DECMAL   00              01              02
;85
;86      ;RPPR     00
;87
;88      ;REI      00                              01
;89
;90      ;QUEUE    00              01      02      03
;91
;92      ;PROBE    00                              01
;93
;94      ;MOVPR    00                              01
;95
;96      ;11MODE   00                              01
;97
;98      ;IE       00                              01
;99
;100     ;MMFW     00                              01
;101
;102     ;ERCODE   00
;103
;104     ;CNSL     00                              01
;105
;106     ;CALCRC                                   00
;107
;108     ;FPA      00                              01
;109
;110     ;GANDF                                    00
;111     .LIST
```

111; This page intentionally left blank.

```
;112     .TOC    "DOC.MIC"
;113     .TOC    "Revision 1.1"
;114     ;        P. R. Guilbault
;115
;116     .TOC    "       Revision History"
;117
;118     ; 01    Add ESCD two byte op-code charts.
;119     ;       Add compatibility mode opcode chart
;120     ; 00    Put revision history from beginning of project til now in section call past history.
;121     ;       Delete 'How to read microcode...'. Refer to MICRO2 USER'S GUIDE(AA-H531A-TE)
;122
;123     .NOLIST         :Disable listing of PCS code for quickie assemblies
```

```
;124     .TOC    "       Tables and charts     : PSL Chart"
;125
;126
;127
;128
;129
;130     ;       +-----------------------------------------------------------+
;131     ;       |                            PSL                            |
;132     ;       +-+-+---+-+-+---+---+-+-----------+-----------+-+-+-+-+-----+
;133     ;       |3|3|2 2|2|2|2|2 2|2 2|2|2 1 1 1 1|1 1 1 1 1 1 0 0|0|0|0|0|0 0 0 0|
;134     ;       |1|0|9 8|7|6|5|5 4|3 2|1|0 9 8 7 6|5 4 3 2 1 0 9 8|7|6|5|4|3 2 1 0|
;135     ;       +-+-+---+-+-+---+---+-+-----------+-----------+-+-+-+-+-----+
;136     ;       |C|T|M  |F|I|C  |P  |M|           |           |D|F|I|T|N|Z|V|C  |
;137     ;       |M|P|B  |P|S|U  |R  |B|    IPL    |    MBZ    |V|U|V| | | | |  |
;138     ;       | | |Z  |D| |R  |V  |Z|           |           | | | | | | | |  |
;139     ;       +-+-+---+-+-+---+---+-+-----------+-----------+-+-+-+-+-----+
;140     ;                               |            PSW                      |
;141     ;                               +-------------------------------------+
;142     ;
;143     ;
;144     ;                       BIT             DESCRIPTION
;145     ;
;146     ;                       3:0             Condition Codes
;147     ;
;148     ;                       4               Trace enable
;149     ;
;150     ;                       5               Integer Overflow trap enable
;151     ;
;152     ;                       6               Floating Underflow trap enable
;153     ;
;154     ;                       7               Decimal Overflow trap enable
;155     ;
;156     ;                       15:8            Reserved to Digital, must be zero
;157     ;
;158     ;                       20:16           Interrupt Priority Level
;159     ;
;160     ;                       21              Reserved to Digital, must be zero
;161     ;
;162     ;                       22:23           Previous Mode
;163     ;
;164     ;                       25:24           Current Mode
;165     ;
;166     ;                       26              Interrupt Stack
;167     ;
;168     ;                       27              First Part Done
;169     ;
;170     ;                       29:28           Reserved to Digital, must be zero
;171     ;
;172     ;                       30              Trace Pending
;173     ;
;174     ;                       31              Compatibility Mode
```

```
;175    .TOC    ""      Tables and charts     : Instruction Name vs Op Code""
;176
;177
;178
;179
;180
```

;181	;	ACBB	9D	BGTR	14	CASEW	AF	CVTLP	F9	FFS	EA	MOVQ	7D	resvrd	59
;182	;	ACBD	6F	BGTRU	1A	CHME	BD	CVTLW	F7	HALT	00	MOVTC	2E	resvrd	5A
;183	;	ACBF	4F	BICB2	8A	CHMK	BC	CVTPL	36	INCB	96	MOVTUC	2F	resvrd	5B
;184	;	ACBL	F1	BICB3	8B	CHMS	BE	CVTTP	26	INCL	D6	MOVW	B0	resvrd	77
;185	;	ACBW	3D	BICL2	CA	CHMU	BF	CVTPT	24	INCW	B6	MOVZBL	9A	RET	04
;186	;	ADAWI	58	BICL3	CB	CLRB	94	CVTPS	08	INDEX	0A	MOVZBW	9B	ROTL	9C
;187	;	ADDB2	80	BICPSW	B9	CLRD	7C	CVTRDL	6B	INSQHI	5C	MOVZWL	3C	RSB	05
;188	;	ADDB3	81	BICW2	AA	CLRF	D4	CVTRFL	4B	INSQTI	5D	MTPR	DA	SBWC	D9
;189	;	ADDD2	60	BICW3	AB	CLRL	D4	CVTSP	09	INSQUE	0E	MULB2	84	SCAN	2A
;190	;	ADDD3	61	BISB2	88	CLRQ	7C	CVTWB	33	INSV	F0	MULB3	85	SKPC	3B
;191	;	ADDF2	40	BISB3	89	CLRW	B4	CVTW	6D	JMP	17	MULD2	64	SOBGEQ	F4
;192	;	ADDF3	41	BISL2	C8	CMPB	91	CVTWF	4D	JSB	16	MULD3	65	SOBGTR	F5
;193	;	ADDL2	C0	BISL3	C9	CMPC3	29	CVTWL	32	LDPCTX	06	MULF2	44	SPANC	2B
;194	;	ADDL3	C1	BISPSW	B8	CMPC5	2D	DECB	97	LOCC	3A	MULF3	45	SUBB2	82
;195	;	ADDP4	20	BISW2	A8	CMPD	71	DECL	D7	MATCHC	39	MULL2	C4	SUBB3	83
;196	;	ADDP6	21	BISW3	A9	CMPF	51	DECW	B7	MCOMB	92	MULL3	C5	SUBD2	62
;197	;	ADDW2	A0	BITB	93	CMPL	D1	DIVB2	86	MCOML	D2	MULP	25	SUBD3	63
;198	;	ADDW3	A1	BITL	D3	CMPP3	35	DIVB3	87	MCOMW	B2	MULW2	A4	SUBF2	42
;199	;	ADWC	D8	BITW	B3	CMPP4	37	DIVD2	66	MFPR	DB	MULW3	A5	SUBF3	43
;200	;	AOBLEQ	F3	BLBC	E9	CMPV	EC	DIVD3	67	MNEGB	8E	NOP	01	SUBL2	C2
;201	;	AOBLSS	F2	BLBS	E8	CMPW	B1	DIVF2	46	MNEGD	72	POLYD	75	SUBL3	C3
;202	;	ASHL	78	BLEQ	15	CMPZV	ED	DIVF3	47	MNEGF	52	POLYF	55	SUBP4	22
;203	;	ASHP	F8	BLEQU	1B	CRC	0B	DIVL2	C6	MNEGL	CE	POPR	BA	SUBP6	23
;204	;	ASHQ	79	BLSS	19	CVTBD	6C	DIVL3	C7	MNEGW	AE	PROBER	0C	SUBW2	A2
;205	;	BB	E1	BLSSU	1F	CVTBF	4C	DIVP	27	MOVAB	9E	PROBEW	0D	SUBW3	A3
;206	;	BBCC	E5	BNEQ	12	CVTBL	98	DIVW2	A6	MOVAD	7E	PUSHAB	9F	SVPCTX	07
;207	;	BBCCI	E7	BNEQU	12	CVTBW	99	DIVW3	A7	MOVAF	DE	PUSHAD	7F	TSTB	95
;208	;	BBCS	E3	BPT	03	CVTDB	68	EDITPC	38	MOVAL	DE	PUSHAF	DF	TSTD	73
;209	;	BBS	E0	BRB	11	CVTDF	76	EDIV	7B	MOVAQ	7E	PUSHAL	DF	TSTF	53
;210	;	BBSC	E4	BRW	31	CVTDL	6A	EMODD	74	MOVAW	3E	PUSHAQ	7F	TSTL	D5
;211	;	BBSS	E2	BSBB	10	CVTDW	69	EMODF	54	MOVB	90	PUSHAW	3F	TSTW	B5
;212	;	BBSSI	E6	BSBW	30	CVTFB	48	EMUL	7A	MOVC3	28	PUSHL	DD	XFC	FC
;213	;	BCC	1E	BVC	1C	CVTFD	56	ESCD	FD	MOVC5	2C	PUSHR	BB	XORB2	8C
;214	;	BCS	1F	BVS	1D	CVTFL	4A	ESCE	FE	MOVD	70	REI	02	XORB3	8D
;215	;	BEQL	13	CALLG	FA	CVTFW	49	ESCF	FF	MOVF	50	REMQHI	5E	XORL2	CC
;216	;	BEQLU	13	CALLS	FB	CVTLB	F6	EXTV	EE	MOVL	D0	REMQTI	5F	XORL3	CD
;217	;	BGEQ	18	CASE	8F	CVTLD	6E	EXTZV	EF	MOVP	34	REMQUE	0F	XORW2	AC
;218	;	BGEQU	1E	CASEL	CF	CVTLF	4E	FFC	EB	MOVPSL	DC	resvrd	57	XORW3	AD

;219 .TOC "" Tables and charts : ESCD Instruction Name vs Op Code(Two byte op codes FDxx)""
;220
;221
;222
;223
;224

;225	;	ACBG	4F	EMODH	74	resvrd	15	resvrd	3C	resvrd	93	resvrd	BA	resvrd	DF
;226	;	ACBH	6F	MNEGG	52	resvrd	16	resvrd	3D	resvrd	94	resvrd	BB	resvrd	E0
;227	;	ADDG2	40	MNEGH	72	resvrd	17	resvrd	3E	resvrd	95	resvrd	BC	resvrd	E1
;228	;	ADDG3	41	MOVAH	7E	resvrd	18	resvrd	3F	resvrd	96	resvrd	BD	resvrd	E2
;229	;	ADDH2	60	MOVAO	7E	resvrd	19	resvrd	57	resvrd	97	resvrd	BE	resvrd	E3
;230	;	ADDH3	61	MOVG	50	resvrd	1A	resvrd	58	resvrd	9A	resvrd	BF	resvrd	E4
;231	;	CLRH	7C	MOVH	70	resvrd	1B	resvrd	59	resvrd	9B	resvrd	C0	resvrd	E5
;232	;	CLRO	7C	MOVO	7D	resvrd	1C	resvrd	5A	resvrd	9C	resvrd	C1	resvrd	E6
;233	;	CMPG	51	MULG2	44	resvrd	1D	resvrd	5B	resvrd	9D	resvrd	C2	resvrd	E7
;234	;	CMPH	71	MULG3	45	resvrd	1E	resvrd	5C	resvrd	9E	resvrd	C3	resvrd	E8
;235	;	CVTBG	4C	MULH2	64	resvrd	1F	resvrd	5D	resvrd	9F	resvrd	C4	resvrd	E9
;236	;	CVTBH	6C	MULH3	65	resvrd	20	resvrd	5E	resvrd	A0	resvrd	C5	resvrd	EA
;237	;	CVTDH	32	POLYG	55	resvrd	21	resvrd	5F	resvrd	A1	resvrd	C6	resvrd	EB
;238	;	CVTFG	99	POLYH	75	resvrd	22	resvrd	77	resvrd	A2	resvrd	C7	resvrd	EC
;239	;	CVTFH	98	PUSHAH	7F	resvrd	23	resvrd	78	resvrd	A3	resvrd	C8	resvrd	ED
;240	;	CVTGB	48	PUSHAO	7F	resvrd	24	resvrd	79	resvrd	A4	resvrd	C9	resvrd	EE
;241	;	CVTGF	33	resvrd	00	resvrd	25	resvrd	7A	resvrd	A5	resvrd	CA	resvrd	EF
;242	;	CVTGH	56	resvrd	01	resvrd	26	resvrd	7B	resvrd	A6	resvrd	CB	resvrd	F0
;243	;	CVTGL	4A	resvrd	02	resvrd	27	resvrd	80	resvrd	A7	resvrd	CC	resvrd	F1
;244	;	CVTGW	49	resvrd	03	resvrd	28	resvrd	81	resvrd	A8	resvrd	CD	resvrd	F2
;245	;	CVTHB	68	resvrd	04	resvrd	29	resvrd	82	resvrd	A9	resvrd	CE	resvrd	F3
;246	;	CVTHD	F7	resvrd	05	resvrd	2A	resvrd	83	resvrd	AA	resvrd	CF	resvrd	F4
;247	;	CVTHF	F6	resvrd	06	resvrd	2B	resvrd	84	resvrd	AB	resvrd	D0	resvrd	F5
;248	;	CVTHG	76	resvrd	07	resvrd	2C	resvrd	85	resvrd	AC	resvrd	D1	resvrd	F8
;249	;	CVTHL	6A	resvrd	08	resvrd	2D	resvrd	86	resvrd	AD	resvrd	D2	resvrd	F9
;250	;	CVTHW	69	resvrd	09	resvrd	2E	resvrd	87	resvrd	AE	resvrd	D3	resvrd	FA
;251	;	CVTLG	4E	resvrd	0A	resvrd	2F	resvrd	88	resvrd	AF	resvrd	D4	resvrd	FB
;252	;	CVTLH	6E	resvrd	0B	resvrd	30	resvrd	89	resvrd	B0	resvrd	D5	resvrd	FC
;253	;	CVTRGL	4B	resvrd	0C	resvrd	31	resvrd	8A	resvrd	B1	resvrd	D6	resvrd	FD
;254	;	CVTRHL	6B	resvrd	0D	resvrd	34	resvrd	8B	resvrd	B2	resvrd	D7	resvrd	FE
;255	;	CVTWG	4D	resvrd	0E	resvrd	35	resvrd	8C	resvrd	B3	resvrd	D8	resvrd	FF
;256	;	CVTWH	6D	resvrd	0F	resvrd	36	resvrd	8D	resvrd	B4	resvrd	D9	SUBG3	43
;257	;	DIVG2	46	resvrd	10	resvrd	37	resvrd	8E	resvrd	B5	resvrd	DA	SUBG2	42
;258	;	DIVG3	47	resvrd	11	resvrd	38	resvrd	8F	resvrd	B6	resvrd	DB	SUBH2	62
;259	;	DIVH2	66	resvrd	12	resvrd	39	resvrd	90	resvrd	B7	resvrd	DC	SUBH3	63
;260	;	DIVH3	67	resvrd	13	resvrd	3A	resvrd	91	resvrd	B8	resvrd	DD	TSTG	53
;261	;	EMODG	54	resvrd	14	resvrd	3B	resvrd	92	resvrd	B9	resvrd	DE	TSTH	73

```
:262    .TOC    ""      Tables and charts     : Op Code vs Instruction Name""
:263
:264
:265
:266
:267
```

:268	:	00	HALT	22	SUBP4	48	CVTFB	6E	CVTLD	91	CMPB	B7	DECW	DC	MOVPSL
:269	:	01	NOP	23	SUBP6	49	CVTFW	6F	ACBD	92	MCOMB	B8	BISPSW	DD	PUSHL
:270	:	02	REI	24	CVTPT	4A	CVTFL	70	MOVD	93	BITB	B9	BICPSW	DE	MOVAF
:271	:	03	BPT	25	MULP	4B	CVTRFL	71	CMPD	94	CLRB	BA	POPR	DE	MOVAL
:272	:	04	RET	26	CVTTP	4C	CVTBF	72	MNEGD	95	TSTB	BB	PUSHR	DF	PUSHAF
:273	:	05	RSB	27	DIVP	4D	CVTWF	73	TSTD	96	INCB	BC	CHMK	DF	PUSHAL
:274	:	06	LDPCTX	28	MOVC3	4E	CVTLF	74	EMODD	97	DECB	BD	CHME	E0	BBS
:275	:	07	SVPCTX	29	CMPC3	4F	ACBF	75	POLYD	98	CVTBL	BE	CHMS	E1	BBC
:276	:	08	CVTPS	2A	SCANC	50	MOVF	76	CVTDF	99	CVTBW	BF	CHMU	E2	BBSS
:277	:	09	CVTSP	2B	SPANC	51	CMPF	77	resvrd	9A	MOVZBL	C0	ADDL2	E3	BBCS
:278	:	0A	INDEX	2C	MOVC5	52	MNEGF	78	ASHL	9B	MOVZBW	C1	ADDL3	E4	BBSC
:279	:	0B	CRC	2D	CMPC5	53	TSTF	79	ASHQ	9C	ROTL	C2	SUBL2	E5	BBCC
:280	:	0C	PROBER	2E	MOVTC	54	EMODF	7A	EMUL	9D	ACBB	C3	SUBL3	E6	BBSSI
:281	:	0D	PROBEW	2F	MOVTUC	55	POLYF	7B	EDIV	9E	MOVAB	C4	MULL2	E7	BBCCI
:282	:	0E	INSQUE	30	BSBW	56	CVTFD	7C	CLRD	9F	PUSHAB	C5	MULL3	E8	BLBS
:283	:	0F	REMQUE	31	BRW	57	resvrd	7C	CLRQ	A0	ADDW2	C6	DIVL2	E9	BLBC
:284	:	10	BSBB	32	CVTWL	58	ADAWI	7D	MOVQ	A1	ADDW3	C7	DIVL3	EA	FFS
:285	:	11	BRB	33	CVTWB	59	resvrd	7E	MOVAD	A2	SUBW2	C8	BISL2	EB	FFC
:286	:	12	BNEQ	34	MOVP	5A	resvrd	7E	MOVAQ	A3	SUBW3	C9	BISL3	EC	CMPV
:287	:	12	BNEQU	35	CMPP3	5B	resvrd	7F	PUSHAD	A4	MULW2	CA	BICL2	ED	CMPZV
:288	:	13	BEQL	36	CVTPL	5C	INSQHI	7F	PUSHAQ	A5	MULW3	CB	BICL3	EE	EXTV
:289	:	13	BEQLU	37	CMPP4	5D	INSQTI	80	ADDB2	A6	DIVW2	CC	XORL2	EF	EXTZV
:290	:	14	BGTR	38	EDITPC	5E	REMQHI	81	ADDB3	A7	DIVW3	CD	XORL3	F0	INSV
:291	:	15	BLEQ	39	MATCHC	5F	REMQTI	82	SUBB2	A8	BISW2	CE	MNEGL	F1	ACBL
:292	:	16	JSB	3A	LOCC	60	ADDD2	83	SUBB3	A9	BISW3	CF	CASEL	F2	AOBLSS
:293	:	17	JMP	3B	SKPC	61	ADDD3	84	MULB2	AA	BICW2	D0	MOVL	F3	AOBLEQ
:294	:	18	BGEQ	3C	MOVZWL	62	SUBD2	85	MULB3	AB	BICW3	D1	CMPL	F4	SOBGEQ
:295	:	19	BLSS	3D	ACBW	63	SUBD3	86	DIVB2	AC	XORW2	D2	MCOML	F5	SOBGTR
:296	:	1A	BGTRU	3E	MOVAW	64	MULD2	87	DIVB3	AD	XORW3	D3	BITL	F6	CVTLB
:297	:	1B	BLEQU	3F	PUSHAW	65	MULD3	88	BISB2	AE	MNEGW	D4	CLRF	F7	CVTLW
:298	:	1C	BVC	40	ADDF2	66	DIVD2	89	BISB3	AF	CASEW	D4	CLRL	F8	ASHP
:299	:	1D	BVS	41	ADDF3	67	DIVD3	8A	BICB2	B0	MOVW	D5	TSTL	F9	CVTLP
:300	:	1E	BCC	42	SUBF2	68	CVTDB	8B	BICB3	B1	CMPW	D6	INCL	FA	CALLG
:301	:	1E	BGEQU	43	SUBF3	69	CVTDW	8C	XORB2	B2	MCOMW	D7	DECL	FB	CALLS
:302	:	1F	BCS	44	MULF2	6A	CVTDL	8D	XORB3	B3	BITW	D8	ADWC	FC	XFC
:303	:	1F	BLSSU	45	MULF3	6B	CVTRDL	8E	MNEGB	B4	CLRW	D9	SBWC	FD	ESCD
:304	:	20	ADDP4	46	DIVF2	6C	CVTLB	8F	CASEB	B5	TSTW	DA	MTPR	FE	ESCE
:305	:	21	ADDP6	47	DIVF3	6D	CVTWD	90	MOVB	B6	INCW	DB	MFPR	FF	ESCF

```
;306     .TOC    ""      Tables and charts     : ESCD Op Code vs Instruction Name(Two byte op codes FDxx)"
;307
;308
;309
;310
;311
;312     ;  00 resvrd      25 resvrd      4A CVTGL       6F ACBH        91 resvrd      B6 resvrd      DB  resvrd
;313     ;  01 resvrd      26 resvrd      4B CVTRGL      70 MOVH        92 resvrd      B7 resvrd      DC  resvrd
;314     ;  02 resvrd      27 resvrd      4C CVTBG       71 CMPH        93 resvrd      B8 resvrd      DD  resvrd
;315     ;  03 resvrd      28 resvrd      4D CVTWG       72 MNEGH       94 resvrd      B9 resvrd      DE  resvrd
;316     ;  04 resvrd      29 resvrd      4E CVTLG       73 TSTH        95 resvrd      BA resvrd      DF  resvrd
;317     ;  05 resvrd      2A resvrd      4F ACBG        74 EMODH       96 resvrd      BB resvrd      E0  resvrd
;318     ;  06 resvrd      2B resvrd      50 MOVG        75 POLYH       97 resvrd      BC resvrd      E1  resvrd
;319     ;  07 resvrd      2C resvrd      51 CMPG        76 CVTHG       98 CVTFH       BD resvrd      E2  resvrd
;320     ;  08 resvrd      2D resvrd      52 MNEGG       77 resvrd      99 CVTFG       BE resvrd      E3  resvrd
;321     ;  09 resvrd      2E resvrd      53 TSTG        78 resvrd      9A resvrd      BF resvrd      E4  res rd
;322     ;  0A resvrd      2F resvrd      54 EMODG       79 resvrd      9B resvrd      C0 resvrd      E5  res rd
;323     ;  0B resvrd      30 resvrd      55 POLYG       7A resvrd      9C resvrd      C1 resvrd      E6  resvrd
;324     ;  0C resvrd      31 resvrd      56 CVTGH       7B resvrd      9D resvrd      C2 resvrd      E7  resvrd
;325     ;  0D resvrd      32 CVTDH       57 resvrd      7C CLRH        9E resvrd      C3 resvrd      E8  resvrd
;326     ;  0E resvrd      33 CVTGF       58 resvrd      7C CLRO        9F resvrd      C4 resvrd      E9  resvrd
;327     ;  0F resvrd      34 resvrd      59 resvrd      7D MOVO        A0 resvrd      C5 resvrd      EA  resvrd
;328     ;  10 resvrd      35 resvrd      5A resvrd      7E MOVAH       A1 resvrd      C6 resvrd      EB  resvrd
;329     ;  11 resvrd      36 resvrd      5B resvrd      7E MOVAO       A2 resvrd      C7 resvrd      EC  resvrd
;330     ;  12 resvrd      37 resvrd      5C resvrd      7F PUSHAH      A3 resvrd      C8 resvrd      ED  resvrd
;331     ;  13 resvrd      38 resvrd      5D resvrd      7F PUSHAO      A4 resvrd      C9 resvrd      EE  resvrd
;332     ;  14 resvrd      39 resvrd      5E resvrd      80 resvrd      A5 resvrd      CA resvrd      EF  resvrd
;333     ;  15 resvrd      3A resvrd      5F resvrd      81 resvrd      A6 resvrd      CB resvrd      F0  resvrd
;334     ;  16 resvrd      3B resvrd      60 ADDH2       82 resvrd      A7 resvrd      CC resvrd      F1  resvrd
;335     ;  17 resvrd      3C resvrd      61 ADDH3       83 resvrd      A8 resvrd      CD resvrd      F2  resvrd
;336     ;  18 resvrd      3D resvrd      62 SUBH2       84 resvrd      A9 resvrd      CE resvrd      F3  resvrd
;337     ;  19 resvrd      3E resvrd      63 SUBH3       85 resvrd      AA resvrd      CF resvrd      F4  resvrd
;338     ;  1A resvrd      3F resvrd      64 MULH2       86 resvrd      AB resvrd      D0 resvrd      F5  resvrd
;339     ;  1B resvrd      40 ADDG2       65 MULH3       87 resvrd      AC resvrd      D1 resvrd      F6 CVTHF
;340     ;  1C resvrd      41 ADDG3       66 DIVH2       88 resvrd      AD resvrd      D2 resvrd      F7 CVTHD
;341     ;  1D resvrd      42 SUBG2       67 DIVH3       89 resvrd      AE resvrd      D3 resvrd      F8  resvrd
;342     ;  1E resvrd      43 SUBG3       68 CVTHB       8A resvrd      AF resvrd      D4 resvrd      F9  resvrd
;343     ;  1F resvrd      44 MULG2       69 CVTHW       8B resvrd      B0 resvrd      D5 resvrd      FA  resvrd
;344     ;  20 resvrd      45 MULG3       6A CVTHL       8C resvrd      B1 resvrd      D6 resvrd      FB  resvrd
;345     ;  21 resvrd      46 DIVG2       6B CVTRHL      8D resvrd      B2 resvrd      D7 resvrd      FC  resvrd
;346     ;  22 resvrd      47 DIVG3       6C CVTBH       8E resvrd      B3 resvrd      D8 resvrd      FD  resvrd
;347     ;  23 resvrd      48 CVTGB       6D CVTWH       8F resvrd      B4 resvrd      D9 resvrd      FE  resvrd
;348     ;  24 resvrd      49 CVTGW       6E CVTLH       90 resvrd      B5 resvrd      DA resvrd      FF  resvrd
```

:349 .TOC "" Tables and charts : Compatibility mode opcode chart"

16-Bit Opcode	Legal Instruc.	Faulting Instructions	16-Bit Opcode	Legal Instruc.	Faulting Instructions
000000		HALT (Rsvrd inst fault)	050000-057777	BIS	
000001		WAIT (Rsvrd inst fault)	060000-067777	ADD	
000002	RTI		070000-070777	MUL	
000003		BPT (BPT inst fault)	071000-071777	DIV	
000004		IOT (IOT inst fault)	072000-072777	ASH	
000005		RESET (Rsvrd inst fault)	073000-073777	ASHC	
000006	RTT		074000-074777	XOR	
000007		MFPT (Rsvrd inst fault)	075000-075007		FADD (Rsvrd inst fault)
000010-000077		Unused (Rsvrd inst fault)	075010-075017		FSUB (Rsvrd inst fault)
000100-000107		JMP (Illegal inst fault)	075020-075027		FMUL (Rsvrd inst fault)
000110-000177	JMP		075030-075037		FDIV (Rsvrd inst fault)
000200-000207	RTS		075040-075777		Unused (Rsvrd inst fault)
000210-000227		Unused (Rsvrd inst fault)	076000-076777		XTD INS (Rsvrd inst fault)
000230-000237		SPL (Rsvrd inst fault)	077000-077777	SOB	
000240	NOP		100000-100377	BPL	
000241-000257	CLR CC'S		100400-100777	BMI	
000260-000277	SET CC'S		101000-101377	BHI	
000300-000377	SWAB		101400-101777	BLOS	
000400-000777	BR		102000-102377	BVC	
001000-001377	BNE		102400-102777	BVS	
001400-001777	BEQ		103000-103377	BCC,BHIS	
002000-002377	BGE		103400-103777	BCS,BLO	
002400-002777	BLT		104000-104377		EMT (BPT inst fault)
003000-003377	BGT		104000-104777		TRAP (TRAP inst fault)
003400-003777	BLE		105000-105077	CLRB	
004x00-004x07		JSR (Illegal inst fault)	105100-105177	COMB	
004x10-004x77	JSR		105200-105277	INCB	
005000-005077	CLR		105300-105377	DECB	
005100-005177	COM		105400-105477	NEGB	
005200-005277	INC		105500-105577	ADCB	
005300-005377	DEC		105600-105677	SBCB	
005400-005477	NEG		105700-105777	TSTB	
005500-005577	ADC		106000-106077	RORB	
005600-005677	SBC		106100-106177	ROLB	
005700-005777	TST		106200-106277	ASRB	
006000-006077	ROR		106300-106377	ASLB	
006100-006177	ROL		106400-106477		MTPS (Rsvrd inst fault)
006200-006277	ASR		106500-106577	MFPD	
006300-006377	ASL		106600-106677	MTPD	
006400-006477		MARK (Rsvrd inst fault)	106700-106777		MFPS (Rsvrd inst fault)
006500-006577	MFPI		107000-107777		Unused (Rsvrd inst fault)
006600-006677	MTPI		110000-117777	MOVB	
006700-006777	SXT		120000-127777	CMPB	
007000-007077		CSM (Rsvrd inst fault)	130000-137777	BITB	
007100-007777		Unused (Rsvrd inst fault)	140000-147777	BICB	
010000-017777	MOV		150000-157777	BISB	
020000-027777	CMP		160000-167777	SUB	
030000-037777	BIT		170000-177777		FLOAT (Rsvrd inst fault)
040000-047777	BIC				

```
:404    .TOC    ""      Tables and charts      : Opcodes and Mnemonics"
:405
:406    ; hex   name    operands                        hex   name    operands
:407    ;-----+-------+-------------------------------+-----+-------+-------------------------
:408    ; 00    HALT                                    20    ADDP4   RW, AB, RW, AB
:409    ; 01    NOP                                     21    ADDP6   RW, AB, RW, AB, RW, AB
:410    ; 02    REI                                     22    SUBP4   RW, AB, RW, AB
:411    ; 03    BPT                                     23    SUBP6   RW, AB, RW, AB, RW, AB
:412    ; 04    RET                                     24    CVTPT   RW, AB, AB, RW, AB
:413    ; 05    RSB                                     25    MULP    RW, AB, RW, AB, RW, AB
:414    ; 06    LDPCTX                                  26    CVTTP   RW, AB, AB, RW, AB
:415    ; 07    SVPCTX                                  27    DIVP    RW, AB, RW, AB, RW, AB
:416
:417    ; 08    CVTPS   RW, AB, RW, AB                  28    MOVC3   RW, AB, AB
:418    ; 09    CVTSP   RW, AB, RW, AB                  29    CMPC3   RW, AB, AB
:419    ; 0A    INDEX   RL, RL, RL, RL, RL, WL          2A    SCANC   RW, AB, AB, RB
:420    ; 0B    CRC     AB, RL, RW, AB, WL              2B    SPANC   RW, AB, AB, RB
:421    ; 0C    PROBER  RB, RW, AB                      2C    MOVC5   RW, AB, RB, RW, AB
:422    ; 0D    PROBEW  RB, RW, AB                      2D    CMPC5   RW, AB, RB, RW, AB
:423    ; 0E    INSQUE  AB, AB                          2E    MOVTC   RW, AB, RB, AB, RW, AB
:424    ; 0F    REMQUE  AB, WL                          2F    MOVTUC  RW, AB, RB, AB, RW, AB
:425
:426    ; 10    BSBB    BB                              30    BSBW    BW
:427    ; 11    BRB     BB                              31    BRW     BW
:428    ; 12    BNEQ    BB                              32    CVTWL   RW, WL
:429    ; 13    BEQL    BB                              33    CVTWB   RW, WB
:430    ; 14    BGTR    BB                              34    MOVP    RW, AB, AB
:431    ; 15    BLEQ    BB                              35    CMPP3   RW, AB, AB
:432    ; 16    JSB     AB                              36    CVTPL   RW, AB, WL
:433    ; 17    JMP     AB                              37    CMPP4   RW, AB, RW, AB
:434
:435    ; 18    BGEQ    BB                              38    EDITPC  RW, AB, AB, AB
:436    ; 19    BLSS    BB                              39    MATCHC  RW, AB, RW, AB
:437    ; 1A    BGTRU   BB                              3A    LOCC    RB, RW, AB
:438    ; 1B    BLEQU   BB                              3B    SKPC    RB, RW, AB
:439    ; 1C    BVC     BB                              3C    MOVZWL  RW, WL
:440    ; 1D    BVS     BB                              3D    ACBW    RW, RW, MW, BW
:441    ; 1E    BGEQU   BB                              3E    MOVAW   AW, WL
:442    ; 1F    BLSSU   BB                              3F    PUSHAW  AW
```

	hex	name	operands	hex	name	operands
;443						
;444						
;445	40	ADDF2	RF, MF	60	ADDD2	RD, MD
;446	41	ADDF3	RF, RF, WF	61	ADDD3	RD, RD, WD
;447	42	SUBF2	RF, MF	62	SUBD2	RD, MD
;448	43	SUBF3	RF, RF, WF	63	SUBD3	RD, RD, WD
;449	44	MULF2	RF, MF	64	MULD2	RD, MD
;450	45	MULF3	RF, RF, WF	65	MULD3	RD, RD, WD
;451	46	DIVF2	RF, MF	66	DIVD2	RD, MD
;452	47	DIVF3	RF, RF, WF	67	DIVD3	RD, RD, WD
;453						
;454	48	CVTFB	RF, WB	68	CVTDB	RD, WB
;455	49	CVTFW	RF, WW	69	CVTDW	RD, WW
;456	4A	CVTFL	RF, WL	6A	CVTDL	RD, WL
;457	4B	CVTRFL	RF, WL	6B	CVTRDL	RD, WL
;458	4C	CVTBF	RB, WF	6C	CVTBD	RB, WD
;459	4D	CVTWF	RW, WF	6D	CVTWD	RW, WD
;460	4E	CVTLF	RL, WF	6E	CVTLD	RL, WD
;461	4F	ACBF	RF, RF, MF, BW	6F	ACBD	RD, RD, MD, BW
;462						
;463	50	MOVF	RF, WF	70	MOVD	RD, WD
;464	51	CMPF	RF, RF	71	CMPD	RD, RD
;465	52	MNEGF	RF, WF	72	MNEGD	RD, WD
;466	53	TSTF	RF	73	TSTD	RD
;467	54	EMODF	RF, RB, RF, WL, WF	74	EMODD	RD, RB, RD, WL, WD
;468	55	POLYF	RF, RW, AB	75	POLYD	RD, RW, AB
;469	56	CVTFD	RF, WD	76	CVTDF	RD, WF
;470	57	* RESERVED TO DEC *		77	* RESERVED TO DEC *	
;471						
;472	58	ADAWI	RW, MW	78	ASHL	RB, RL, WL
;473	59	* RESERVED TO DEC *		79	ASHQ	RB, RQ, WQ
;474	5A	* RESERVED TO DEC *		7A	EMUL	RL, RL, RL, WQ
;475	5B	* RESERVED TO DEC *		7B	EDIV	RL, RQ, WL, WL
;476	5C	INSQHI	AB,WQ	7C	CLRQ	WQ
;477	5D	INSQTI	AB,AQ	7D	MOVQ	RQ, WQ
;478	5E	REMQHI	AQ,WL	7E	MOVAQ	AQ, WL
;479	5F	REMQTI	AQ,WL	7F	PUSHAQ	AQ

	hex	name	operands	hex	name	operands
;480						
;481						
;482	80	ADDB2	RB, MB	A0	ADDW2	RW, MW
;483	81	ADDB3	RB, RB, WB	A1	ADDW3	RW, RW, WW
;484	82	SUBB2	RB, MB	A2	SUBW2	RW, MW
;485	83	SUBB3	RB, RB, WB	A3	SUBW3	RW, RW, WW
;486	84	MULB2	RB, MB	A4	MULW2	RW, MW
;487	85	MULB3	RB, RB, WB	A5	MULW3	RW, RW, WW
;488	86	DIVB2	RB, MB	A6	DIVW2	RW, MW
;489	87	DIVB3	RB, RB, WB	A7	DIVW3	RW, RW, WW
;490						
;491	88	BISB2	RB, MB	A8	BISW2	RW, MW
;492	89	BISB3	RB, RB, WB	A9	BISW3	RW, RW, WW
;493	8A	BICB2	RB, MB	AA	BICW2	RW, MW
;494	8B	BICB3	RB, RB, WB	AB	BICW3	RW, RW, WW
;495	8C	XORB2	RB, MB	AC	XORW2	RW, MW
;496	8D	XORB3	RB, RB, WB	AD	XORW3	RW, RW, WW
;497	8E	MNEGB	RB, WB	AE	MNEGW	RW, WW
;498	8F	CASEB	RB, RB, RB	AF	CASEW	RW, RW, RW
;499						
;500	90	MOVB	RB, WB	B0	MOVW	RW, WW
;501	91	CMPB	RB, RB	B1	CMPW	RW, RW
;502	92	MCOMB	RB, WB	B2	MCOMW	RW, WW
;503	93	BITB	RB, RB	B3	BITW	RW, RW
;504	94	CLRB	WB	B4	CLRW	WW
;505	95	TSTB	RB	B5	TSTW	RW
;506	96	INCB	MB	B6	INCW	MW
;507	97	DECB	MB	B7	DECW	MW
;508						
;509	98	CVTBL	RB, WL	B8	BISPSW	RW
;510	99	CVTBW	RB, WW	B9	BICPSW	RW
;511	9A	MOVZBL	RB, WL	BA	POPR	RW
;512	9B	MOVZBW	RB, WW	BB	PUSHR	RW
;513	9C	ROTL	RB, RL, WL	BC	CHMK	RW
;514	9D	ACBB	RB, RB, MB, BW	BD	CHME	RW
;515	9E	MOVAB	AB, WL	BE	CHMS	RW
;516	9F	PUSHAB	AB	BF	CHMU	RW

	hex	name	operands	hex	name	operands
;519	C0	ADDL2	RL, ML	E0	BBS	RL, VB, BB
;520	C1	ADDL3	RL, RL, WL	E1	BBC	RL, VB, BB
;521	C2	SUBL2	RL, ML	E2	BBSS	RL, VB, BB
;522	C3	SUBL3	RL, RL, WL	E3	BBCS	RL, VB, BB
;523	C4	MULL2	RL, ML	E4	BBSC	RL, VB, BB
;524	C5	MULL3	RL, RL, WL	E5	BBCC	RL, VB, BB
;525	C6	DIVL2	RL, ML	E6	BBSSI	RL, VB, BB
;526	C7	DIVL3	RL, RL, WL	E7	BBCCI	RL, VB, BB
;527						
;528	C8	BISL2	RL, ML	E8	BLBS	RL, BB
;529	C9	BISL3	RL, RL, WL	E9	BLBC	RL, BB
;530	CA	BICL2	RL, ML	EA	FFS	RL, RB, VB, WL
;531	CB	BICL3	RL, RL, WL	EB	FFC	RL, RB, VB, WL
;532	CC	XORL2	RL, ML	EC	CMPV	RL, RB, VB, RL
;533	CD	XORL3	RL, RL, WL	ED	CMPZV	RL, RB, VB, RL
;534	CE	MNEGL	RL, WL	EE	EXTV	RL, RB, VB, WL
;535	CF	CASEL	RL, RL, RL	EF	EXTZV	RL, RB, VB, WL
;536						
;537	D0	MOVL	RL, WL	F0	INSV	RL, RL, RB, VB
;538	D1	CMPL	RL, RL	F1	ACBL	RL, RL, ML, BW
;539	D2	MCOML	RL, WL	F2	AOBLSS	RL, ML, BB
;540	D3	BITL	RL, RL	F3	AOBLEQ	RL, ML, BB
;541	D4	CLRL	WL	F4	SOBGEQ	ML, BB
;542	D5	TSTL	RL	F5	SOBGTR	ML, BB
;543	D6	INCL	ML	F6	CVTLB	RL, WB
;544	D7	DECL	ML	F7	CVTLW	RL, WW
;545						
;546	D8	ADWC	RL, ML	F8	ASHP	RB, RW, AB, RB, RW, AB
;547	D9	SBWC	RL, ML	F9	CVTLP	RL, RW, AB
;548	DA	MTPR	RL, RL	FA	CALLG	AB, AB
;549	DB	MFPR	RL, WL	FB	CALLS	RL, AB
;550	DC	MOVPSL	WL	FC	XFC	
;551	DD	PUSHL	RL	FD	ESCD	* RESERVED TO DEC *
;552	DE	MOVAL	AL, WL	FE	ESCE	* RESERVED TO DEC *
;553	DF	PUSHAL	AL	FF	ESCF	* RESERVED TO DEC *

```
;554    .TOC    ""      Tables and charts     : Operand Specifier Codes"
;555
;556    ; Hex   Name                                R M W A V Index
;557    ;-------------------------------------------------------
;558    ; 0-3   Literal                             y f f f f   f
;559    ;  4    Indexed                             y y y y y   f
;560    ;  5    Register                            y y y f y   f
;561    ;  6    Register deferred                   y y y y y   y
;562    ;  7    Autodecrement                       y y y y y   ux
;563    ;  8    Autoincrement                       y y y y y   ux
;564    ;  9    Autoincrement deferred              y y y y y   ux
;565    ;  A    Byte Displacement                   y y y y y   y
;566    ;  B    Byte Displacement deferred          y y y y y   y
;567    ;  C    Word Displacement                   y y y y y   y
;568    ;  D    Word Displacement deferred          y y y y y   y
;569    ;  E    Longword Displacement               y y y y y   y
;570    ;  F    Longword Displacement deferred      y y y y y   y
;571
;572    ;Floating Literals
;573
;574    ;       Hex  Value      Hex  Value      Hex  Value      Hex  Value
;575    ;       ---------      ---------      ---------      ---------
;576    ;       00 - 1/2        01 - 9/16       02 - 5/8        03 - 11/16
;577    ;       04 - 3/4        05 - 13/16      06 - 7/8        07 - 15/16
;578    ;       08 - 1          09 - 1 1/8      0A - 1 1/4      0B - 1 3/8
;579    ;       0C - 1 1/2      0D - 1 5/8      0E - 1 3/4      0F - 1 7/8
;580    ;       10 - 2          11 - 2 1/4      12 - 2 1/2      13 - 2 3/4
;581    ;       14 - 3          15 - 3 1/4      16 - 3 1/2      17 - 3 3/4
;582    ;       18 - 4          19 - 4 1/2      1A - 5          1B - 5 1/2
;583    ;       1C - 6          1D - 6 1/2      1E - 7          1F - 7 1/2
;584    ;       20 - 8          21 - 9          22 - 10         23 - 11
;585    ;       24 - 12         25 - 13         26 - 14         27 - 15
;586    ;       28 - 16         29 - 18         2A - 20         2B - 22
;587    ;       2C - 24         2D - 26         2E - 28         2F - 30
;588    ;       30 - 32         31 - 36         32 - 40         33 - 44
;589    ;       34 - 48         35 - 52         36 - 56         37 - 60
;590    ;       38 - 64         39 - 72         3A - 80         3B - 88
;591    ;       3C - 96         3D - 104        3E - 112        3F - 120
```

```
;592     .TOC    "       Tables and charts      : Machine Check Error Logout"
;593
;594     ;       At any machine check, the error handling microcode attempts to logout
;595     ;  the following information.  Ordinarily, it appears on the stack as shown,
;596     ;  but if a double error halt occurs, the operator can find the same inform-
;597     ;  ation in the ID bus temporaries.  This information is STAR-specific, of
;598     ;  course, and does not apply to other members of the family.
;599
;600     ;       Data            Memory loc'n    ID loc'n        Notes
;601     ;       ------------------------------------------------------------------
;602     ;       Byte Count      (SP)            none            40(dec) = 28(hex)
;603     ;       Summary Param   (SP)+4          T0 (30)         See below
;604     ;       CPU Error Status(SP)+8          T1 (31)         See CES register format
;605     ;       Trapped UPC     (SP)+12         T2 (32)         Microcode error loc'n
;606     ;       VA/VIBA         (SP)+16         T3 (33)         Virtual address
;607     ;       D register      (SP)+20         T4 (34)
;608     ;       TB ERR 0        (SP)+24         T5 (35)         See TBER0 format
;609     ;       TB ERR 1        (SP)+28         T6 (36)         See TBER1 format
;610     ;       Timeout Addr    (SP)+32         T7 (37)         Physical addr/4
;611     ;       Parity          (SP)+36         T8 (38)         See PARITY format
;612     ;       SBI Error       (SP)+40         T9 (39)         See SBI.ERR format
;613     ;       PC              (SP)+44         none
;614     ;       PSL             (SP)+48         none
;615
;616     ;       The summary parameter is a longword.  Byte 1 is a flag, which is
;617     ;  non-zero if a CP timeout or CP error confirmation interrupt was pending
;618     ;  at the time the machine check occurred.  The interrupt, if any, has been
;619     ;  cleared.  Byte zero identifies the type of machine check:
;620
;621     ;       00 - CP Read Timeout or Error Confirmation Fault
;622     ;       02 - CP Translation Buffer Parity Error Fault
;623     ;       03 - CP Cache Parity Error Fault
;624     ;       05 - CP Read Data Substitute Fault
;625     ;       0A - IB Translation Buffer Parity Error Fault
;626     ;       0C - IB Read Data Substitute Fault
;627     ;       0D - IB Read Timeout or Error Confirmation Fault
;628     ;       0F - IB Cache Parity Error Fault
;629     ;       F1 - Control Store Parity Error Abort
;630     ;       F2 - CP Translation Buffer Parity Error Abort
;631     ;       F3 - CP Cache Parity Error Abort
;632     ;       F4 - CP Read Timeout or Error Confirmation Abort
;633     ;       F5 - CP Read Data Substitute Abort
;634     ;       F6 - Microcode "not supposed to get here" abort
;635
;636     ;       "IB" above refers to memory reads generated by the instruction
;637     ;  buffer in the process of prefetching the instruction stream.  In these
;638     ;  cases, the address stored at (SP)+16 is from VIBA.  "CP" refers
;639     ;  to memory references explicitly requested by microcode and whose
;640     ;  address comes from VA.
```

```
;641     .TOC    ""      Tables and charts     : Fixed Address Allocation''
;642
;643     ; 0000  001F    AFORK Specifier Evaluation
;644     ; 0020  003C    Service
;645     ; 0040  004F    AFORK Instruction Entries (including FPD)
;646     ; 0080  008F    AFORK Instruction Entries (including FPD)
;647     ; 00C0  00CF    AFORK Instruction Entries (including FPD)
;648     ; 00FF          HALT Loop
;649     ;
;650     ; 0100  010F    Microtrap Vectors
;651     ; 0120  012F    Console Service Entries
;652     ;
;653     ; 0200  021F    BFORK Specifier Evaluation
;654     ; 0220  022F    BFORK R-mode Execution Entries
;655     ; 0240  024F    BFORK Instruction Entries
;656     ; 0280  028F    BFORK Instruction Entries
;657     ; 02C0  02CF    BFORK Instruction Entries
;658     ;
;659     ; 0300  031F    SPEC/WPD Specifier Evaluation
;660     ; 0340  034F    CFORK Instruction Entries
;661     ; 0380  038F    CFORK Instruction Entries
;662     ; 03C0  03CF    CFORK Instruction Entries
;663     ;
;664     ; 0400  041F    ASPC Specifier Evaluation
;665     ; 0440  044F    DFORK Instruction Entries
;666     ; 0480  048F    DFORK Instruction Entries
;667     ; 04C0  04CF    DFORK Instruction Entries
;668     ;
;669     ; 0700  07FF    Compatability mode Entries
;670     ;
;671     ;WCS Fixed Addresses
;672     ;
;673     ; 1000  102F    Accelerator AFORK traps
;674     ; 1040  104F    Accelerator AFORK traps
;675     ; 1080  108F    Accelerator AFORK traps
;676     ; 10C0  10CF    Accelerator AFORK traps
;677     ; 10E0          WCS Entry for code 2 vectors
;678     ; 1100  110F    WCS uTRAP Entries
;679     ; 1120  112F    WCS Debugger
;680     ; 1140  117F    uECO Entries
;681     ;
;682     ; 1200  122F    Accelerator BFORK traps
;683     ; 1240  124F    Accelerator BFORK traps
;684     ; 1280  128F    Accelerator BFORK traps
;685     ; 12C0  12CF    Accelerator BFORK traps
;686     ;
;687     ; 1340  134F    Accelerator CFORK traps
;688     ; 1380  138F    Accelerator CFORK traps
;689     ; 13C0  13CF    Accelerator CFORK traps
```

```
;690     .TOC    "       Tables and charts     : Control Store Field Map"
;691
;692     ;       *                *                *                *                *
;693     ;       | 15  14  13  12 | 11  10  09  08 | 07  06  05  04 | 03  02  01  00 |
;694     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;695     ;       |    EALU       |                JMP               |               |
;696     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;697
;698
;699     ;       *                *                *                *                *
;700     ;       | 31  30  29  28 | 27  26  25  24 | 23  22  21  20 | 19  18  17  16 |
;701     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;702     ;       |  IEK  |       MSC       |VAK|FEK|SCK|     CCK     |  EBMX |  SMX  |
;703     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;704
;705
;706     ;       *                *                *                *                *
;707     ;       | 47  46  45  44 | 43  42  41  40 | 39  38  37  36 | 35  34  33  32 |
;708     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;709     ;       |ADS|    MCT/CID    | FS|           SPO             |    PCK    |
;710     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;711
;712
;713     ;       *                *                *                *                *
;714     ;       | 63  62  61  60 | 59  58  57  56 | 55  54  53  52 | 51  50  49  48 |
;715     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;716     ;       |          MPA          |  SI/ACM   |       QK      |    SGN    |
;717     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;718
;719
;720     ;       *                *                *                *                *
;721     ;       | 79  78  77  76 | 75  74  73  72 | 71  70  69  68 | 67  66  65  64 |
;722     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;723     ;       |   DT  |RMX|       BEN         |  ACF  |      ALU      |  SUB  |
;724     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;725
;726
;727     ;       *                *                *                *                *
;728     ;       | 95  94  93  92 | 91  90  89  88 | 87  86  85  84 | 83  82  81  80 |
;729     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
;730     ;       |      IBC      |       DK      |    SHF    |    BMX    |  AMX  |
;731     ;       *---+---+---+---*---+---+---+---*---+---+---+---*---+---+---+---*
```

;732 .TOC "" Tables and charts : Branch Enable Functions"
;733
;734 ;BEN; Name | UPC<02> | UPC<01> | UPC<00>
;735 ;---+---+---+---+---
;736 ; 0 | NOP | 0 | 0 | 0
;737 ;---+---+---+---+---
;738 ; 1 | Z | 0 | 0 | ALU Z
;739 ;---+---+---+---+---
;740 ; 2 | ROR | LA<01> | PSL<C> | LA<00>
;741 ;---+---+---+---+---
;742 ; - | C31 | 0 | ALU C31 | 0
;743 ;---+---+---+---+---
;744 ; 4 | | | | |
;745 ;---+---+---+---+---
;746 ; 5 | | | | |
;747 ;---+---+---+---+---
;748 ; 6 | ACCelerator | ACC UB2 | ACC UB1 | ACC UB0
;749 ;---+---+---+---+---
;750 ; 7 | | | | |
;751 ;---+---+---+---+---
;752 ; 8 | DATA TYPE 0= Normal 2= Field Src | Read + | ASRC + | ASRC +
;753 ;VAX; 1= Q + D 3= Addr Src | Modify | VSRC | Q + D
;754 ;---+---+---+---+---
;755 ; 8 | END DP1 | Read + | O Class | J Class +
;756 ;-11; | Modify | | DM27
;757 ;---+---+---+---+---
;758 ; 9 | IR2-1 | 0 | IR<2> | IR<1>
;759 ;VAX; | | | |
;760 ;---+---+---+---+---
;761 ; 9 | PC Modes | 0 | SM or DM | Dst R
;762 ;-11; | | | 47 + 57 | .eq. PC
;763 ;---+---+---+---+---
;764 ; A | REI | Mode .lt. | | |
;765 ; | | ASTLVL | | |
;766 ;---+---+---+---+---
;767 ; B | IB TEST 0= IB Miss 2= Stall | 0 | IB running | IB ERROR +
;768 ; | 1= Error 3= Data OK | | | DATA VALID
;769 ;---+---+---+---+---
;770 ; C | MUL | SC.ne.0 | D<01> | D<00>
;771 ;---+---+---+---+---
;772 ; D | SIGNS | Q<31> | D.ne.0 | D<31>
;773 ;---+---+---+---+---
;774 ; E | INTERRUPT | AC Low | Internal | Interrupt
;775 ; | | | Interrupt | Request
;776 ;---+---+---+---+---
;777 ; F | Decimal | 0 | D<7:0> | D<3:0>=
;778 ; | | | 30-39 | 0B + 0D
;779 ;---+---+---+---+---

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

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

```
;825    .TOC    ""      Tables and charts     : Memory Control Functions"      ;October 11, 1976
;826
;827    ;                          ---  trap on  ---
;828    ;           |  |  F|   |  |T A       O     S|   Y = utrap on condition
;829    ;           |  |  U|  C|  |B C X A T D T   B|   * = utrap on condition unless MSC/
;830    ;           |  |  N|  H|S |M C P L B D B C I|       SECOND.REF or RETRY.NO.TRAP
;831    ; A         |V |  C|  E|A |I E A I U A P P E|   N = do not utrap on condition
;832    ; D  MCT F  |/ |  T|  C|V |S S G G F D A A R|   - = hardware behaviour undefined.
;833    ; S 3210 S  |P |  N|  K|E |S S E N M R R R R|       ucode must prevent condition
;834    ; ---------+--+---+---+--+-----------------+---------- ----------
;835    ; 0 0000 0  |V |   |  R|N |N N N N N N Y N N| TEST.RCHK
;836    ; 0 0001 0  |V |   |   |N |N N N N N N N N N| MEM.NOP
;837    ; 0 0010 0  |V |   |  W|N |N N N N N N Y N N| TEST.WCHK
;838    ; 0 0011 0  |  |   |   |  |                 |
;839    ;
;840    ; 0 0100 0  |  |   |   |  |                 |
;841    ; 0 0101 0  |V |  W|   |N |Y N - - N - Y N Y| WRITE.V.NOCHK
;842    ; 0 0110 0  |V |  W|  W|Y |Y Y * * Y Y Y Y Y| WRITE.V.WCHK
;843    ; 0 0111 0  |V |IW |   |N |- - - - - - Y N Y| LOCKWRITE.V.XCHK
;844    ;
;845    ; 0 1000 0  |V |  R|  R|Y |Y Y * * N Y Y Y Y| READ.V.RCHK
;846    ; 0 1001 0  |V |  R|   |N |Y N - - N - Y Y Y| READ.V.NOCHK
;847    ; 0 1010 0  |V |  R|  W|Y |Y Y * * Y Y Y Y Y| READ.V.WCHK
;848    ; 0 1011 0  |V |  R| IB|Y |Y Y Y Y Y Y Y Y Y| READ.V.IBCHK
;849    ;
;850    ; 0 1100 0  |V |  R|  R|N |N N N N N Y N N N| READ.V.NEWPC
;851    ; 0 1101 0  |V | IR|   |N |Y N - - N - Y Y Y| LOCKREAD.V.NOCHK
;852    ; 0 1110 0  |V | IR|  W|Y |Y Y - - Y - Y Y Y| LOCKREAD.V.WCHK
;853    ; 0 1111 0  |  |   |   |  |                 |
;854    ;
;855    ; 1 0000 0  |  |HOLD |N |N N N N N N N N N| SBI.HOLD
;856    ; 1 0001 0  |  |UNJAM|N |N N N N N N N N N| SBI.HOLD+UNJAM
;857    ; 1 0010 0  |P |INVAL|N |N N N N N N N N N| INVALIDATE
;858    ; 1 0011 0  |P | VAL |N |N N N N N N N N N| VALIDATE
;859    ;
;860    ; 1 0100 0  |P |EXTWR|N |N N N N N N N N Y| EXTWRITE.P
;861    ; 1 0101 0  |P |  W  |N |N N N N N N N N Y| WRITE.P
;862    ; 1 0110 0  |  |     |  |                 |
;863    ; 1 0111 0  |P |  IW |N |N N N N N N N N Y| LOCKWRITE.P
;864    ;
;865    ; 1 1000 0  |  |     |  |                 |
;866    ; 1 1001 0  |P |  R  |N |N N N N N N N Y Y| READ.P
;867    ; 1 1010 0  |  |     |  |                 |
;868    ; 1 1011 0  |P | ISR |N |N N N N N N N N Y| READ.INT.SUM
;869    ;
;870    ; 1 1100 0  |  |     |  |                 |
;871    ; 1 1101 0  |P |  IR |N |N N N N N N N Y Y| LOCKREAD.P
;872    ; 1 1110 0  |  |     |  |                 |
;873    ; 1 1111 0  |I |  R  |N |N N N N N N N N N| ALLOW.IB.READ
;874    ;
;875    ; 0 XXXX 1  |  |     |  |N N N N N N N N N| NO MEMORY OPERATION
;876    ; 1 XXXX 1  |I |  R  |N |N N N N N N N N N| DEFAULT: ALLOW IB READ
;877    ;
;878    ; Abort Ref on Trap? A A A   A A A R A (A=any, R=read)
```

```
;879     .TOC     ""       Tables and charts      : ID Bus Map''     ;September 2, 1977
;880
;881     ;  The left column lists first the ID bus register name, and in parenthesis
;882     ; its address.  If the register is also accessible by MTPR and MFPR inst-
;883     ; ructions, the Internal Register Number and symbolic name are also given.
;884
;885     ;            31/15  30/14  29/13  28/12  27/11  26/10  25/09  24/08  23/07  22/06  21/05  20/04  19/03  18/02  17/01  16/00
;886     ;IBUF      *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;887     ;(00)      |                    Data Byte 3                    |                    Data Byte 2                    |
;888     ;          |  31  |  30  |  29  |  28  |  27  |  26  |  25  |  24  |  23  |  22  |  21  |  20  |  19  |  18  |  17  |  16  |
;889     ;          *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;890     ;          |                    Data Byte 1                    |                    Data Byte 0                    |
;891     ;          |  15  |  14  |  13  |  12  |  11  |  10  |  9   |  8   |  7   |  6   |  5   |  4   |  3   |  2   |  1   |  0   |
;892     ;          *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;893     ;          *                    *                    *                    *                    *                    *
;894     ;DAY.TIM   *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;895     ;(01)      |                    Time Byte 3                    |                    Time Byte 2                    |
;896     ;          |  31  |  30  |  29  |  28  |  27  |  26  |  25  |  24  |  23  |  22  |  21  |  20  |  19  |  18  |  17  |  16  |
;897     ;IR 1B     *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;898     ;''TODR''  |                    Time Byte 1                    |                    Time Byte 0                    |
;899     ;          |  15  |  14  |  13  |  12  |  11  |  10  |  9   |  8   |  7   |  6   |  5   |  4   |  3   |  2   |  1   |  0   |
;900     ;          *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;901     ;          *                    *                    *                    *                    *                    *
;902     ;SYS.ID    *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;903     ;(03)      |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |
;904     ;          |  31  |  30  |  29  |  28  |  27  |  26  |  25  |  24  |  23  |  22  |  21  |  20  |  19  |  18  |  17  |  16  |
;905     ;IR 3E     *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;906     ;''SID''   |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |  ID  |
;907     ;          |  15  |  14  |  13  |  12  |  11  |  10  |  9   |  8   |  7   |  6   |  5   |  4   |  3   |  2   |  1   |  0   |
;908     ;          *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;909     ;          *                    *                    *                    *                    *                    *
;910     ;RXCS      *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;911     ;(04)      |                                                                                                             |
;912     ;          |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   |
;913     ;IR 20     *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;914     ;''RXCS''  |                                                         |Intrpt|                                             |
;915     ;          |  0   |  0   |  0   |  0   |  0   |  0   |  0   |  0   | Done |Enable|  0   |  0   |  0   |  0   |  0   |  0   |
;916     ;          *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;917     ;          *                    *                    *                    *                    *                    *
;918     ;RXDB      *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;919     ;(05)      |                    Data Byte 3                    |                    Data Byte 2                    |
;920     ;          |  31  |  30  |  29  |  28  |  27  |  26  |  25  |  24  |  23  |  22  |  21  |  20  |  19  |  18  |  17  |  16  |
;921     ;IR 21     *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
;922     ;''RXDB''  |                    Data Byte 1                    |                    Data Byte 0                    |
;923     ;          |  15  |  14  |  13  |  12  |  11  |  10  |  9   |  8   |  7   |  6   |  5   |  4   |  3   |  2   |  1   |  0   |
;924     ;          *------+------+------+------+------+------+------+------+------+------+------+------+------+------+------+------*
```

	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
TXCS (06)																
IR 22 "TXCS"	0	0	0	0	0	0	0	0	Ready	Intrpt Enable	0	0	0	0	0	0
TXDB (07)	Data Byte 3											Data Byte 2				
IR 23 "TXDB"	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Data Byte 1											Data Byte 0				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DQ (08)	D(read) Q(write)															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	D(read) Q(write)															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NXT.PER (09)	Next Interval															
IR 19 "NICR"	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Next Interval															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLK.CS (0A)																
IR 18 "ICCS"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Error	0	0	0	0	0	0	0	Intrpt Req	Intrpt Enable	Single Clock	Xfer	0	0	0	Run
INTERVAL (0B)	Count (microseconds)															
IR 1A "ICR"	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Count (microseconds)															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
CES (0C)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Nested Error
IR 13 "ASTR" IR 3D "PME"	Control Summry	Store 2	Parity 1	Error 0	EALU N	EALU Z	ALU N	ALU Z	ALU C31	Arithmetic Trap Code			Perf Mon En	AST Level		
VECTOR (0D)	0	0	0	0	0	0	Prior Valid	Priority				Number of Ones				
	0	0	0	0	0	0	0	08	07	06	05	Vector 04	03	02	01	00
SIR (0E)	0	0	0	0	0	0	0	0	0	0	0	Interrupt Priority Level Active				
IR 15 "SISR"	Software Interrupt Register															0
	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	
PSL (0F)	Compat Mode	Trace Pend	0	0	FPD	Intrpt Stack	Current Mode		Previous Mode		0	Interrupt Priority Level				
IR 12 "IPL"	0	0	0	0	0	0	0	0	Decmal oVrflo	Float Undflo	Intger oVrflo	T	N	Condition Code Z	V	C
TBUF (10)	Valid	Protection Code			Modify	0	0	0	0	0	0	20	Page Frame Number 19	18	17	16
	Page Frame Number															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
TBER0 (12)												Replac	Force	Replace	Force	TB Miss
	0	0	0	0	0	0	0	0	0	0	0	Both	G1	G0	G1	G0
				Last Reference					TB Hit				Force TB			Mem
	FS	ADS	MCT3	MCT2	MCT1	MCT0	IBWCHK	AR	G1	G0	0		Parity Error			Man En
TBER1 (13)													TB Parity Error Bits			
	0	0	0	0	0	0	0	0	0	0	0	1D2	1D1	1D0	0D2	0D1
			TB Parity Error Bits					CP TB		Last		Bad		IPA		
	0D0	1A2	1A1	1A0	0A2	0A1	0A0	PE	0	TB WrP	0	IPA	Miss	PE	Prot E	Auto L
ACC.MN (16)	Write															
	Trp Ad	0	0	0	0	0	0	0					Trap Address			
	Write	Micro														
	uBreak	Match	0	0	0	0	0						Micro-break (write) Current Address (read)			
ACC.CS (17)					Resrvd											
	Error	0	0	0	Oprand	0	0	0	0	0	0	0	0	0	0	0
IR 28 "ACCS"	Accel															
	Enable	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SILO (18)	After	SBI			SBI ID				SBI TAG			SBI			SBI	
	Fault	Intlk	4	3	2	1	0	2	1	0	M3/B31	M2/B30	M1/B29	M0/B28	CNF1	CNF0
IR 31 "SBIS"								SBI TR								
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBI.ERR (19)																
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IR 34 "SBIER"	RDS			CP	CP TO	CP TO		CP Err	IB	IB	IB TO	IB TO	IB Err	Mult	Not	
	Int En	CRD	RDS	TO	ST1	ST0	0	CNF	RDS	TO	ST1	ST0	CNF	Error	Busy	0

	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
TIME.ADR (1A)	Mode		Prot		Physical Address											
	1	0	Check	0	29	28	27	26	25	24	23	22	21	20	19	18
IR 35 "SBITA"	Physical Address															
	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
FAULT (1B)	Parity Fault	0	Unexp RD	0	Mult Xmit	Xmit Fault	0	0	0	0	0	0	Fault Latch	Fault Int En	Fault Signal	Fault Lock
IR 30 "SBIFS"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COMP (1C)	Silo Lock	Silo Int En	Lock Uncond	Cond Lock Code		Compare Command or Mask				Compare Tag			Count			
IR 32 "SBISC"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MAINT (1D)	Rev P0	Wr Seq Fault	Unexp RD	Mult Xmit	Maintenance ID 4	3	2	1	0	Force SBI	Enable Invalid	Reverse Cache Parity				Force MissG0
IR 33 "SBIMT"	Force MissG1	Force Rep G0	Force Rep G1	Disabl SBI	Rev P1	G1 Match	G0 Match	Force TO	0	0	0	0	0	0	0	0
PARITY (1E)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Any Error	CP Error	Data Parity OK G1 B0	G1 B1	G1 B2	G1 B3	G0 B0	G0 B1	G0 B2	G0 B3	G0	Address Parity OK G0 B1	G0 B2	G1 B0	G1 B1	G1 B2

	31/15	30/14	29/13	28/12	27/11	26/10	25/09	24/08	23/07	22/06	21/05	20/04	19/03	18/02	17/01	16/00
USTACK (20)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				12	11	10	9	8	Control Store Address 7	6	5	4	3	2	1	0
UBREAK (21)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IR 3C 'MBRK'	0	0	0	12	11	10	9	8	Control Store Address 7	6	5	4	3	2	1	0
WCS.ADDR (22)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IR 2C 'WCSA'	Invert Parity	Mod 3 Counter		12	11	10	9	8	Control Store Address 7	6	5	4	3	2	1	0
WCS.DATA (23)	Data 31	Data 30	Data 29	Data 28	Data 27	Data 26	Data 25	Data 24	Data 23	Data 22	Data 21	Data 20	Data 19	Data 18	Data 17	Data 16
IR 2D 'WCSD'	Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0

Scratch Pad Locations

Name	Addr	IR #	Symb	Name	Addr	IR #	Symb
P0BR	24	08	P0BR	T2	32		
P1BR	25	0A	P1BR	T3	33		
SBR	26	0C	SBR	T4	34		
KSP	28	00	KSP	T5	35		
ESP	29	01	ESP	T6	36		
SSP	2A	02	SSP	T7	37		
USP	2B	03	USP	T8	38		
ISP	2C	04	ISP	T9	39		
FPDA	2D			PCBB	3A	10	PCBB
D.SV	2E			SCBB	3B	11	SCBB
Q.SV	2F			P0LR	3C	09	P0LR
T0	30			P1LR	3D	0B	P1LR
T1	31			SLR	3E	0D	SLR

```
;1151   .TOC    ''      Tables and charts     : Past history''
;1152
;1153   ;WCS120 USE FPLA VERSION 0C.
;1154   ;       ADD 4 NEW INTERLOCKED QUEUE INSTRUCTIONS.
;1155   ;       ADD PLA TERM FOR G AND H FLOATING
;1156   ;       FIX CVTPL PRECISION ON OVERFLOW
;1157   ;       FIX CVTPS V BIT
;1158   ;       CHANGE C FORK TO HANDLE QUAD LITERALS FOR EMODD
;1159   ;       ADD TEST FOR RESERVED OPERAND IN ASHP
;1160   ;       FIX ACB INTEGER WHERE COMPARE CAUSES OVERFLOW
;1161   ;       FINX INDEX HANG UP WITH ZERO SIZE AND NEGATIVE INDEX
;1162   ;       CHANGE BRANCH ON BIT TO WORK WITH NEGATIVE POSITION
;1163   ;       FIX CMPP BUG WHERE SECOND STRING IS LONGER
;1164   ;       CHANGE INSQUE CONTEXT FOR SETTING CONDITION CODES
;1165   ;       FIX INSV WHEN DEST IS ACCROSS REG BOUNDRY
;1166   ;       FIX MULP SETTING OF Z BIT
;1167   ;       CHANGE RET TO NOT CLEAR T BIT WHEN (FP) IS TB MISS
;1168   ;       FIX ADDP6/4 SETTING OF V BIT WHEN DEST IS SHORTER
;1169   ;       THAN SRC STRING
;1170   ;       FIX DIVD BY ZERO TO STORE THE SECOND LONGWORD OF THE QUOTIENT
;1171   ;       CORRECTLY
;1172   ;       FIX DIVP TO PREVENT HANG UP ON NON DECIMAL DATA
;1173   ;       FIX EMODD TO GIVE CORRECT RESULT WITH FRACTIONS ALL ONES
;1174   ;WCS118 REFIX TO MOVTC/MOVTUC HANDLING OF INTERRUPTS TO PREVENT HANG.
;1175   ;WCS117 USE FPLA VERSION 0A.
;1176   ;       SRM CHANGE TO EDITPC DEFINITION OF REPLACE SIGN TO IGNORE PSW<N>.
;1177   ;       FIX REPORTING OF PC ON ACCESS VIOLATION FAULT WHEN READ REFERENCE
;1178   ;       CROSSES PAGE BOUNDARY AND SECOND PAGE IS KNOWN TO TB BUT INACCESSIBLE.
;1179   ;       SRM CHANGE TO FORCE IPL TO 1F WHEN EXCEPTION VECTOR SELECTS SERVICE
;1180   ;       ON THE INTERRUPT STACK.
;1181   ;       FIX AFORK TREATMENT OF ADDRESS SOURCES WHEN SPECIFIER IS IMMEDIATE.
;1182   ;       FIX BFORK/CFORK HANDLING OF IMMEDIATE SPECIFIERS FOR WRITE DESTS
;1183   ;       TO GIVE RESERVED ADDRESSING MODE INSTEAD OF MACHINE CHECK.
;1184   ;       FIX DEFERED (INDIRECT) ADDRESSING FLOWS TO CHECK FOR INTERRUPTS TO
;1185   ;       HOLD DOWN WORST CASE INTERRUPT LATENCY.
;1186   ;       FIX MOVTC/MOVTUC AND SKPC/LOCC TO TEST FOR INTERRUPTS IN LOOP.
;1187   ;WCS116 USE FPLA VERSION 9.
;1188   ;       FIX ASHP TO DETECT CARRY OUT OF BIT 31 WHEN ROUNDING, NOT BIT 7.
;1189   ;       FIX CVTTP TO STORE CORRECT BYTE OF TRANSLATION TABLE WHEN SOURCE
;1190   ;       LENGTH IS ONE.
;1191   ;WCS115 USE WITH FPLA VERSION 8
;1192   ;       FIX INSV WHICH ACCESSED SECOND ALIGNED LONGWORD IF BIT 31 OF FIRST
;1193   ;       WAS BEING CHANGED. FIX MTPR TO ALLOW GARBAGE IN BITS 31:24 OF LENGTH
;1194   ;       REGISTERS, AND TO DELETE INCORRECT CHECK ON PROCESS BASE REGISTERS.
;1195   ;WCS114.BIN     FIX MEM.MGMT. PROBLEM WITH ADDRESS BITS 31-16 REPORTED
;1196   ;       NOT ZERO ON ACCESS VIOLATION IN COMP.MODE.
;1197   ;WCS113.BIN     GENERAL RELEASE ,GOES WITH FPLA VERSION #4.
;1198   ;       FIX DIVIDE BY ZERO PROBLEM IN DIV.
;1199   ;       FIX CLOCKING OF BYTE-COUNT IN MOVC3.
;1200   ;WCS111.BIN     GENERAL RELEASE , NO NEW FPLA.
;1201   ;       FIX EVALUATION OF SHORT LITERALS IN EMOD.
;1202   ;       FIX PATCH TO ACBF.
;1203   ;       FIX CRC MEMORY OR REGISTER FLAG(DELETE IT)
;1204   ;WCS110.BIN     GENERAL RELEASE FOR FPLA VERSION #3
;1205   ;       ADD ECO TO CLEAR TP-BIT DURING INTERRUPTS IN CHAR.STR.INSTR.
```

```
;1206   ;WCS025 FIX POLYF SGN NOT CLEARED BEFORE BEN
;1207   ;       FIX MOST-NEGATIVE-NUMBER PROBLEM IN EMUL.
;1208   ;WCS024 FIX INTERRUPT/TRAP PROBLEM IN CMP-MODE.
;1209   ;WCS023 FIX INTERRUPT PROBLEMS IN ADDP4,MULP,CVTPT
;1210   ;WCS022 FIX INTERRUPT-PROBLEM IN DIVP.
;1211   ;WCS021 FIX 96TH-BIT ERROR PROBLEM IN MULD, OVERFLOW BUG IN POLYD
;1212   ;WCS020 FIX MEMORY MANAGEMENT BUG.
;1213   ;WCS109.BIN     SUPERCEDES WCS108 FOR GENERAL RELEASE
;1214   ;WCS108.BIN     GENERAL RELEASE FOR FPLA VERSION #2
;1215   ;WCS01B CLEAR Q-REGISTER IN DIVF-ROUTINE TO AVOID ROUNDING-ERROR.
;1216   ;WCS01A ADD TEST FOR RESERVED OPERAND ON LIMIT IN ACBD.
;1217   ;WCS019 FIX OVERFLOW PROBLEMS IN CONVERT FLOATING INSTRUCTIONS.
;1218   ;WCS106.BIN     FOR PLA VERSION #1
;1219   ;WCS018 FIX EMODD BUG WHEN PRODUCT=0 (ONLY STORED HIGH FRACT)
;1220   ;WCS017 FIX CMPD PATCH TO COMPARE MIDDLE FRACTION BITS CORRECTLY
;1221   ;WCS016 CHANGE CRC FOR SRM ECO(4 OPERANDS)
;1222   ;WCS015 PATCH PATCH TO CVTLP IN "PATCH.MIC".
;1223   ;WCS014 PATCH PATCHES TO ACBD AND ACBF.
;1224   ;WCS013 PATCH AN ADDRESS-CONSTRAINT IN "PATCH.MIC" IN ABCD-PATCH.
;1225   ;WCS012 ADD ECO'S TO INSQ AND REMQ-INSTRUCTIONS IN PATCH-FILE.
;1226   ;WCS011 FIX MULL PROBLEM IN FPA MICRO-CODE(TAKING INVALID EXCEPT )
;1227   ;134    FIX RET ECO (#128) TO TREAT MASK AS COMPLEMENTED
;1228   ;133    FIX -0-BUG IN CVTPL.
;1229   ;132    FIX CONDITION CODE SYNC PROBLEM IN FPA INTERFACE FOR POLY
;1230   ;131    FIX BUGS IN EDIV AND CVTDF.
;1231   ;       FIX OVERFLOW-PROBLEM IN CVTPL.
;1232   ;130    FIX POLYD RESTART PROBLEM - R3 GETTING BUMPED PREMATURELY
;1233   ;129    ADD MACROS SO AS TO MAKE PREVIOUS VERSION ASSEMBLE.
;1234   ;128    FIX RET-INSTRUCTION SO AS TO RESTORE R0,R1 (USING PATCH.MIC).
;1235   ;127    FIX CALL TO BUILD REGISTER MASK FOR MPUSH CORRECTLY. SHF DOES NOT
;1236   ;       INSERT 1'S, NO MATTER WHAT IS SHIFT INSERT, ON CONTEXT SHIFTS.
;1237   ;       ADD FIXES TO ACBF AND ACBD IN PATCH.MIC.
;1238   ;126    FIX EMODD FPA CODE ON R-MODE FOR M'CAND(WCS CHANGE ONLY)
;1239   ;       FIX EMOD FOR BOTH BASE AND ACCELERATOR TO SETUP PARAMETERS FOR
;1240   ;       PROBE SUBROUTINE CORRECLTLY.
;1241   ;125    FIX CALLS TO GET.PTE-ROUTINE TO INCLUDE MEMORY NOOPS.
;1242   ;124    FIX MULD FIX TO CORRECT TYPOS.
;1243   ;       ADD PATCHES TO WCS TO FIX BUGS IN ACBF,ACBD,CVTLP,MATCHC.
;1244   ;123    FIX ERROR SYNC PROBLEMS IN FPA POLYF & POLYD CODE(WCS CHANGE ONLY)
;1245   ;122    FIX ACCURACY PROBLEM IN MULD (WCS CHANGE ONLY)
;1246   ;121    FIX CONSTRAINT AT ADDFDX BROKEN BY 120.
;1247   ;120    FIX ADD/SUB FLOATING TO STORE RESULTS EVEN IF SRC=0 (THUS 3-OPERAND
;1248   ;       WORKS, TOO).  FIX DOUBLE-TO-INTEGER CONVERTS TO CALL UNPK CORRECTLY.
;1249   ;       FIX ASHQ AGAIN, TO SET V CORRECTLY IF OVERFLOW DETECTED EARLY
;1250   ;       ON SHIFT BY MULTIPLE OF 32.
;1251   ;119    REPLACE CONSTRAINT DELETED ACCIDENTALLY IN 118 FIX OF CALL.
;1252   ;       RECODE CVTLF TO SAVE 3 STATES    MOST CASES, FIX INTEGER-TO-DOUBLE
;1253   ;       CONVERTS TO BE SMALLER, FAS      D PACK RESULT BEFORE STORING.
;1254   ;118    FIX UNALIGNED STACK BUG IN       , CLEAR FPD IN MOVTUC IF ESCAPE SEEN
;1255   ;117    TIE DOWN EMODF.11 IN WCS SO EDIV CAN CALL IT FOR RESULT STORAGE
;1256   ;116    FIX VARIOUS POLYF/POLYD BUGS, ADD ECO FOR ADDD IN PROM
;1257   ;115    INITIALIZE STATE FOR SRC LEN = 0 CASE OF EDITPC
;1258   ;       FIX SPURIOUS UNDERFLOW DETECTION BUG IN POLYD
;1259   ;114    FIX ASHQ TO CORRECTLY DETECT OVERFLOW ON LEFT SHIFTS BY 32.
;1260   ;113    ADD PATCH TO PROM VERSION OF MUL, EMUL TO FIX NEG MUL BUG
```

```
;1261    ;         ALSO FIX POLYF TO PRODUCE 30-31 BITS OF PRODUCT, NOT 29-30
;1262    ;112      CLEAN-UP DEFINITIONS IN RMAC
;1263    ;         MAKE SPACE AT 1120-112F FOR WCS DEBUGGER ROUTINES
;1264    ;111      INCLUDE UECO.T09,UECO.T10,UECO.J05,UECO.J06,INTRPT.FRM IN PATCH.MIC.
;1265    ;110      FIX ACBF AND ACBD TO BUMP PC OVER THE BDEST CORRECTLY.
;1266    ;109      CORRECTED VERSION OF DIVP SO WCS LOAD DOESN'T INCLUDE ROM ADDRESS 3CD.
;1267    ;         NEW, SUPPOSEDLY CLEANER VERSIONS OF INTEGER MUL/DIV,
;1268    ;         FLOATING AND DOUBLE PRECISION ARITHMETIC ROUTINES INSERTED -
;1269    ;         WATCH THIS SPACE FOR FURTHER DETAILS.
;1270    ;         RECODING OF BB WITH BDEST=0 TO SAVE TIME
;1271    ;         RECODING OF MOVF, MNEGF TO SAVE TIME.
;1272    ;108      ENDFLOAT CHAR CHANGED RESTART IN EDITPC
;1273    ;         NEW POLYF,POLYD
;1274    ;107      FIXED HALF-MAGNITUDE BUG IN CVTFX
;1275    ;         SUBSTITUTED NEW MULD, EMOD TO CONFORM TO NEW EMOD SPEC
;1276    ;106      CHANGE MOVCCMPC5 RETURN TO FIT IN WCS SPACE
;1277    ;105      INCLUDE UECO.J04 IN PATCH.MIC.
;1278    ;         AND CORRECT ADDRESS IN PATCH.MIC FOR INSV.2A, FROM 1A6 TO 0B6.
;1279    ;104      FIX TYPING-ERROR IN PATCH.MIC, CAUSING ERRORS
;1280    ;         IN 'READ-BCD'-ROUTINE.
;1281    ;103      INCLUDE PATCHES IN THE FILES:
;1282    ;         UECO.J01,MOV1.FRM,UECO.J02,UECO.J03,CMP1.FRM,UECO.T04,
;1283    ;         SPAN1.FRM,SKP1.FRM,UECO.T01,UECO.T02,UECO.T03,UECO.T05,
;1284    ;         UECO.T06,UECO.T07,UECO.T08,CVTF.FRM,
;1285    ;         CVTRFL.FRM,CVTDB.FRM,CVTRDL.FRM
;1286    ;         RELAX ADDRESS CONSTRAINTS IN BFORK AND CFORK.
;1287    ;102      CHANGE EMODF TO CONFORM TO NEW ACCURACY SPEC, AND TO
;1288    ;         CAUSE EXCEPTION ON INTEGER OVERFLOWS.
;1289    ;         GENERAL CLEANUP OF SKPC, LOCC, SPANC, SCANC, MATCHC, MACCOM
;1290    ;         EDITPC- SRC LENGTH = 0 CHECK ADDED
;1291    ;101      FIX CONDITIONAL ASSEMBLER DIRECTIVES IN DIVP.
;1292    ;         FIX INSV TO TAKE RESERVED OPERAND ON POSITION .GTRU. 31, NOT
;1293    ;         .GTRU. 255 (POSITION WAS BEING SHIFTED BEFORE TESTING WITH K[.1F]).
;1294    ;100      FIX OVERFLOW IN DIVP.
;1295    ;         CLR FPD IN POLYD FOR DEGREE EQUAL 0.
;1296    ;99       NEW FLOAT/INTEGER CONVERT SUBROUTINES, EMODF CHANGES TO MATCH
;1297    ;         CHANGE POLYF/D TO FLUSH AT END OF INSTRUCTIONS.
;1298    ;         FIX CMPC5 BUG PARALLEL TO MOVC5 BUG BELOW
;1299    ;98       FIX LIMIT FOR COUNTER IN DIVP.
;1300    ;97       EDITPC SRC LEN > 31 FIX
;1301    ;         MOVC5 FILL CHAR DUPLICATION BEFORE REGISTER STORAGE
;1302    ;         FIX POLYF DST SIGN.
;1303    ;96       2 BUG FIXES IN SRC WHEN DEST=REGISTER (THANKS TO PFC)
;1304    ;         FIX POLYF UNDERFLOW TRAP.
;1305    ;         MORE SORDID EDITPC RESTART BUG
;1306    ;         FIX CONDITIONAL ASSEMBLY BUG IN DIVP FOR SPLIT PROM/WCS.
;1307    ;95       ASSORTED EDITPC FPD/RESTART FIXES
;1308    ;94       PRESERVE RLOG WHEN GETTING PC DELTA AND INSERT SUB/SPEC FOR CC IN POLY.
;1309    ;         INITIAL FILL BY MOVE RESTART IN EDITPC DEST ADDR
;1310    ;93       EDITPC MOVE/FLOAT RESTART WRITE LEFT/RIGHT
;1311    ;92       YET ANOTHER EDITPC FIX
;1312    ;         MATCHC-CORRECT OUTER COUNTER = 0 DURING FAULT IN INNER LOOP CASE
;1313    ;91       FIX EDITPC FPD RESTART ADDR FOR READING SIGN NIB
;1314    ;         FIX DIVP RESTART-ROUTINE.
;1315    ;90       MSC FIXES AND CLEAN UP'S IN FLOATING, ARITH, POLY AND ADDD.
```

```
;1316    ;89     FIX RESTART ROUTINE IN DIVP.
;1317    ;       SEVERAL CHANGES IN FLOATING TO INTEGER CONVERSIONS TO FIX PROBLEMS
;1318    ;       WITH ZERO OPERANDS CAUSING UNDERFLOW, AND IN CVTLD IN
;1319    ;       STORING LOW-ORDER PART OF RESULT
;1320    ;88     FIX RTI IN 11-MODE TO USE ONLY BITS 0-4 OF PS ON STACK.
;1321    ;       FIX PROBER TO CATCH ACCESS VIOLATIONS NOT CHECKED BY GETPTE.
;1322    ;       CHANGE REI TO ALLOW RESTORED PSL TO CONTAIN IS=1, IPL.EQL.0.
;1323    ;       THIS IS IN CONFLICT WITH SRM CHAPTER 6, BUT CONSISTENT WITH SIMULATOR.
;1324    ;       CHAPTER 6 AVOIDS THE PROBLEM BY FORCING IPL=1F ON EXCEPTION WHICH
;1325    ;       MUST BE SERVICED ON INTERRUPT STACK.
;1326    ;87     TAKE OUT ARITH TRAP FOR CM DIV BY 0 OR OVERFLOW.
;1327    ;       FIX A CONSTRAINED BLOCK IN POLYF.
;1328    ;       INITIALIZE R15 ON RESTART IN DIVP.
;1329    ;86     FIX EMODF INTEGER REG MODE.
;1330    ;       FIX CM DIV OVFL TO SET PSL <Z>.
;1331    ;85     FIX CM DIV NOT CHECKING THE D'SOR SIGN CORRECTLY AND ZERO EXT D'END <L>.
;1332    ;       FIX CLR.FPD RACE IN MOVT(U)C, ALSO RESTORE ESC CHAR CORRECTLY
;1333    ;       FIX DIVF2 UNDERFLOW TO SET PSL <Z>.
;1334    ;       FIX MOVP TO PREVENT SETTING OF V-BIT.
;1335    ;84     TAKE OUT THE WRONG FIX OF CM SM/DM67 ODD ADDR CHK, RE-LATCH REG'S
;1336    ;       IN CM SM DECODING, FIX TRAP CODE 0 TO HAVE 0 INSTEAD OF SP1.CON.
;1337    ;       FIX PROBE TO CHECK BOTH BASE AND BASE+LEN-1 EVEN IF BASE GETS
;1338    ;       TRANSLATION-NOT-VALID.
;1339    ;83     FIX RESTART ADDRESS FOR CMPP.
;1340    ;       FIX CM FOR SM/DM67 ODD ADDR CHK, ZERO EXT WORD ON VA/PC,
;1341    ;       AND COND CODES FOR SWAB, SAVE SOME DUPLICATED STATES.
;1342    ;82     SAVE A STATE IN CMPP.
;1343    ;       FIX MPUSH ROUTINE TO UPDATE SP AFTER FETCHING REGISTER TO STORE.
;1344    ;       EDITPC FPD/RESTART MECHANISM REARRANGED
;1345    ;       MATCHC FPD/RESTART COUNTER RESTORED AS NEG NUMBER
;1346    ;       FIX CM ASHC TO EXCLUDE LEFT WORD ON SRC REG AND JSR TO WRITE
;1347    ;       WORD ONLY.  FIX CM SM/DM2 PROBLEMS TO READ DT INST DEP.
;1348    ;       FIX CHM CALL SETFPD TO INCLUDE READ FAIL.
;1349    ;81     NEW(RL) MOVC,CMPC,MTC,MTUC,CRC
;1350    ;80     FIX CARRY RESTORE IN ADDP/SUBP-RESTART ROUTINE.
;1351    ;       FIX JSR CM TO WORK ON FAULTS AND NOT MESS GENERAL REG.
;1352    ;       DELETE UNUSED MACROS, AND RE-ARRANGE DEFNEW TO GET FIELDS INTO
;1353    ;       ALPHABETICAL ORDER.
;1354    ;79     MOVC5 CONSTRAINT FIELD FOR SRC LEN = DEST LEN
;1355    ;78     ADD ACBF AND ACBD.
;1356    ;77     CHANGE MEMORY-FAULT RETURN-ADDRESS FOR ADDP AND SUBP TO
;1357    ;       13, AND LOAD IT DIRECTLY IN FPDA FROM KMX.
;1358    ;       REMOVE UNNECESSARY STATES IN AFORK FOR SERVICE CONDITIONS.
;1359    ;       CO-ORDINATE EDITPC RESTART INTERFACES
;1360    ;76     DECREASE FPD SET-UP CODE + HANDLE UNDEFINED RESTART STATES
;1361    ;       FOR EDITPC, MATCHC, SKPC, LOCC, SPANC + SCANC
;1362    ;       MOVC3 STATE REG CORRECTED FOR LONG-BYTE TRANSITION
;1363    ;       FIX MULD ROUTINE DUE TO EMODD CHANGES.
;1364    ;       MAINTAIN PSL <V> FOR DIVD.
;1365    ;75     SET CES ARITH TRAP FOR DIVIDE BY 0.  FIX CM FAULTS TO LOAD PC
;1366    ;       TO RC[PC.SV] FOR CALLING BAKUP.PC RTN.
;1367    ;       REWRITE WPR.10 SUBROUTINE, WHICH CLEARS TB, SO THAT STEP COUNT IN
;1368    ;       SC CAN BE SPECIFIED DIFFERENT BY CALLER WHO ENTERS AT WPR.10B.
;1369    ;       THIS ALLOWS CHANGE TO LDPCTX TO CLEAR ONLY PROCESS SPACE ENTRIES.
;1370    ;74     REMOVE INTERRUPT STROBE IN IRD & IRD.11 STATES -- MUST NOT STROBE
```

```
;1371    ;       WHILE ALSO BRANCHING ON THE RESULT.  FIX INT.B, WHICH BACKS UP
;1372    ;       REGS AND PC FOR INTERRUPT IN MIDDLE OF INSTRUCTION, TO LOAD PC INTO
;1373    ;       RC[PC.SV], WHICH IS WHERE BAKUP.PC EXPECTS TO FIND IT.
;1374    ;       MODIFY WPR.10 SUBROUTINE, WHICH CLEARS TB, TO WORK USING VA31
;1375    ;       AS INDEX INTO TRANSLATION BUFFER INSTEAD OF VA14.
;1376    ;73     CHANGE EXIT FROM INTERRUPT-ROUTINE IN BCD STRING-INSTRUCTIONS.
;1377    ;       ADD TEST FOR ILLEGAL LENGTHS IN MULP.
;1378    ;       CHANGE CM VARIOUS FAULTS TO BACKUP REG'S AND PC BEFORE SERVICE THEM
;1379    ;       IN EXCEPTIONS.
;1380    ;72     STROBE INTERRUPTS IN IRD & IRD.11 STATES, SO INTERRUPTS CAN BE TAKEN
;1381    ;       WHEN STALL OCCURS.
;1382    ;       FIX MULF/DIVF PSL COND CODES PROBLEMS DUE TO LAST CHANGE.
;1383    ;       FIX MOVF, MNEGF, MOVD, MNEGD TO CORRECTLY DETECT EXPONENT OF ZERO FOR
;1384    ;       CLEANING UP DIRTY ZERO.  ALSO COMBINE FLOWS TO SAVE STATES.
;1385    ;       FIX RETURN-ADDRESS AFTER MEMORY-FAULTS IN BCD STRING-INSTRUCTIONS.
;1386    ;       EDITPC MAJOR EDIT
;1387    ;       FIX INTERRUPT CODE FOR MIDDLE OF INSTRUCTIONS TO BACKUP REGISTERS
;1388    ;       IF ENTERED AT INT.B, BUT JUST CLEAR TP IF ENTERED AT INT.I.
;1389    ;71     SAVE A STATE IN ASHP WITH NEGATIVE SHIFT-COUNT.
;1390    ;       SAVE A ROM-LOCATION IN CONSOLE MICRO-CODE BY NOT ALLOWING
;1391    ;       FOR AN INCORRECT BYTE/WORD/LONG PARAMETER IN EXAM. AND DEPOSIT.
;1392    ;       CHANGE ALL INTEGER AND FLOATING ARITH TO TAKE ADVANTAGE OF NEW
;1393    ;       FEATURES OF WRITE DEST, AND FREE "WDST" AND "WDSTF" ROUTINES.
;1394    ;       FREE LOCATIONS ALLOWED FOR A RETURN61 FROM "ASPC" RTN FOR POLY'S.
;1395    ;70     REMOVE LOCATIONS WHICH ALLOWED FOR A RETURN61 FROM
;1396    ;       THE ROUTINE ASPC, WHEN CALLED BY THE DECIMAL STRING-INSTRUCTIONS.
;1397    ;       RELAX CONSTRAINED BLOCKS OF ADDF ROUTINES.  FIX DIVD.
;1398    ;       ADD TEST FOR INTERRUPTS IN DIVP.
;1399    ;69     CHANGE CMPP AND CVTTP TO ALLOW FOR THE FACT THAT IN THIS
;1400    ;       MACHINE NABS(0)=300.
;1401    ;       SET UNDERFLOW WHEN RESULT EXP IS 0 IN FLOATING ARITH.
;1402    ;68     ADD TESTS FOR INTERRUPTS IN CMPP AND MOVP.
;1403    ;       SKPC/LOCC FPD/RESTART LENGTH COUNTER CORRECTED.
;1404    ;       FIX CM SM/DM3,5 TO HAVE 2 INSTEAD OF SP1/2.CON, AND HAVE RLOG
;1405    ;       FOR DM EVALUATION.
;1406    ;       INHIBIT CM ADDRESS FOR VECTOR FETCHING IN INTRPT/EXCEPTION HANDLING.
;1407    ;       SHUFFLE BISPSW/BICPSW TO RELAX TEST CONSTRAINT.
;1408    ;       ADD INTERRUPT TEST IN ASPC TO HELP REDUCE LATENCY.  SIMILAR ENTRY
;1409    ;       IN SPEC CONFLICTS WITH MTPR, SO TURNED OFF AT THE MOMENT.
;1410    ;       ADD CODE AT INT.I TO CLEAR TP ON INTERRUPTS TAKEN IN MIDDLE OF INSTR'S.
;1411    ;67     CONSOLE EXAMINE AND DEPOSIT ROUTINES NOW RETURN PHYSICAL
;1412    ;       ADDRESS EVEN WHEN MEMORY MANAGEMENT IS OFF.
;1413    ;       CONSOLE DEPOSIT ROUTINE NOW CHECKS FOR SBI TIME-OUT,
;1414    ;       AND REPORTS IT TO CONSOLE.
;1415    ;       FIX SRC-LENGTH IN CVTPS, AVOID USE OF RC 7 DURING SPECIFIER
;1416    ;       EVALUATIONS IN BCD STRING INSTRUCTIONS.
;1417    ;       INCLUDE CODE FOR TSTF/D IN BFORK, AND MOVE CODE FOR MOV/MNEG F/D FROM
;1418    ;       DUMMY FILE FLTADR TO BFORK.
;1419    ;66     FIX INTERRUPT HANDLING, SO IT WILL TEST CM MODE CORRECTLY.
;1420    ;64     RELAX ADDRESS CONSTRAINTS IN ASHP AND DIVP, SAVE SOME MEMORY-
;1421    ;       LOCATIONS IN ASHP.
;1422    ;       FIX FFS, FFC FOR SIZE=32 WHEN NO FIND OCCURS.  POSITION SHOULD BE
;1423    ;       UPDATED BY 32, NOT 0.
;1424    ;63     DECIMAL STRING INSTRUCTIONS NOW CLEAR FPD BEFORE THEY FLUSH THE IB.
;1425    ;       INTRUPT AND EXCPT HANDLING WILL CLEAR CES ARITH TRAP CODE.
```

```
;1426    ;       FIX DIVF2 FOR INCREMENTING PC ONE TOO MANY.
;1427    ;       SKPC RESTART COUNTER FIXED SOME MORE(AT SETFPD INTERFACE).
;1428    ;62     ALL STRING INSTRUCTIONS CHECK FOR FINAL DEST REGISTER WRITE
;1429    ;       SKPC RESTART COUNTER CORRECTED
;1430    ;       MOVC3 FORWARD UNALIGNED 1ST BYTE FPD STATE REGISTER SETTING CORRECTED
;1431    ;       DECIMAL DIVIDE BY ZERO NOW HAS TRAP CODE 4.
;1432    ;       CVTLP-INSTRUCTION NOW CHECKS OVERFLOW IN A WAY WHICH SAVES
;1433    ;       FOUR INSTRUCTIONS, BUT COSTS ONE STATE.
;1434    ;       A CHANGE HAS BEEN MADE TO ADDRESS-CONSTRAINTS IN EXAMINE- AND
;1435    ;       DEPOSIT-ROUTNES IN CONSOLE MICRO-CODE.
;1436    ;61     EXTV (AND DERIVATIVES) FIXED FOR REGISTER BASE WHEN SIZE =32.  THIS
;1437    ;       WAS GOING AWRY BECAUSE OF HACK TO FORCE EALU N WHILE GETTING S-32.
;1438    ;       ALSO RELAX SOME ADDRESS CONSTRAINTS IN FIELD CODE.
;1439    ;60     .DEFAULT/CMHALT IS SET TO 0 IN IE.MIC TO DISABLE HALT ON RESERVED
;1440    ;       INSTRUCTIONS IN CM.  SET TO 1 WILL MAKE THESE INSTR HALT.
;1441    ;       CHM SAVES R[SP] BEFORE PROBING AND DOES NOT LOSE PSL <FPD>
;1442    ;       AFTER CALL SET FPD ROUTINE.
;1443    ;       CHANGE "ADDFSH" ROUTINE FOR POLYF.
;1444    ;       SKPC/LOCC RESTART AFTER INTERRUPT CORRECTED.
;1445    ;59     INTERRUPT/EXCEPTION FLAG FIXED IN STRING INSTRUCTIONS
;1446    ;       NEW SPEC FOR CALLS/CALLG TO SAVE PSW WITH TBIT CLEARED.
;1447    ;58     MOVC + CMPC BEN/ALU1-0 CORRECTED
;1448    ;       THE DEPOSIT- AND EXAMINE-ROUTINE OF THE CONSOLE MICRO-CODE,
;1449    ;       NOW RETURN THE PHYSICAL ADDRESS.
;1450    ;       PX-PTE'S OVERLAYING ITS SYS-PTE FIX IS FIXED BY SAVING THE FAULTED
;1451    ;       VIRTUAL ADDRESS IN RC[MBIT.VA] FOR SETTING OF SYS-PTE'S  OVERLOOKED
;1452    ;        IN ORIGINAL FIX.
;1453    ;57     FIX BFORK TO SEND FIRST HALF OF QUAD/DOUBLE MEMORY OPERAND TO
;1454    ;       ACCELERATOR AT B.MQ. INCLUDE DECIMAL INSTRUCTIONS.
;1455    ;       SWAP ADDF (DST MEM MODE), MULF3 C-FORK ENTRIES.
;1456    ;56     FIX ASHQ TO AVOID LOSING TRACK OF SIGNS IN LONG LEFT SHIFT
;1457    ;55     NEW SKPC, LOCC, MOVTC, MOVTUC, SCANC, SPANC, MATCHC,
;1458    ;       CRC, EDIT TO MINIMIZE ADDRESS CONSTRAINTS.
;1459    ;       FIXES OVERLAY PROBLEM OF SYS PTE FOR A P1/P2 PT ON SETTING M-BIT.
;1460    ;54     MODIFY PROBE TO TEST RLOG.EMPTY? ON JUMP TO BAKUP.RGS, WHICH IS THE
;1461    ;       NEW CALLING SEQUENCE.  CHANGE BAKUP.RGS ROUTINE TO AVOID TESTING
;1462    ;       RLOG.EMPTY WHILE READING RLOG, WHICH CAUSES RACE.
;1463    ;       ADD MEM.NOP TO OVERRIDE DEFAULT OF ALLOW.IB.READ AT CRITICAL PLACES
;1464    ;       THROUGHOUT MEMORY MANAGEMENT CODE, TO PREVENT AUTO-RELOAD CYCLES
;1465    ;       WHILE WRITING TB OR TESTING VAMX BITS 31 AND 30.
;1466    ;53     FIX MTPR WHICH INVALIDATES TB TO INHIBIT IB, THEREBY PREVENTING
;1467    ;       AUTO-RELOAD CYCLES FROM CHANGING ADDRESS PRESENTED TO TB.
;1468    ;52     FIX MUL TO SET V CORRECTLY.
;1469    ;       FIX EDIV TO RESET OPERAND ADDR.
;1470    ;51     FIX LDPCTX TO FLUSH TRANSLATION BUFFER.
;1471    ;       FIX CHM FOR GETTING THE RIGHT VECTOR.
;1472    ;50     SET PSL<IS> WHEN EXCEPTION WITH VECTOR [01:00] = 1.
;1473    ;49     FIX A LIKELY TIMING PROBLEM IN EDIV.
;1474    ;       FIX ASHQ TO DETECT OVERFLOW CORRECTLY ON LEFT SHIFTS OF 31, 63, ETC.
;1475    ;48     CORRECT REI TEST OF IPL'S, BROKEN IN 47.
;1476    ;       FIX MULB FOR SETTING V RIGHT, AND ADDF FOR ALU C31 PROBLEM.
;1477    ;       FIX CONSOLE CONSTRAINED BLOCK.
;1478    ;       CHANGE REFERENCES TO MCHECK TO CALL EH.USEQ TO LOG OUT MACHINE
;1479    ;       CHECK WHEN PROCESSOR ATTEMPTS SPECIFIER EVALUATION THAT SHOULD NOT
;1480    ;       HAPPEN ACCORDING TO IRC ROMS.  INCLUDE ADAWI INSTRUCTION.
```

```
;1481    ;47     RESTORE THE CORRECT VERSION OF IE.MIC.
;1482    ;       CHANGES TO PUSHR, POPR, RET, AND REI TO RELAX BRANCH CONSTRAINTS
;1483    ;       FIX ADDF3 TO STORE CORRECTLY, AND MULB2 SET COND CODE N CORRECTLY.
;1484    ;46     INSERT FIELD EXITS TO STOR.L, NOT STORE
;1485    ;       MOVC3/5, CMPC3/5, SPANC, SCANC, MOVTC, MOVTUC, SKPC, LOCC,
;1486    ;       EDITPC, CRC, + MATCHC ALL UPDATED FOR FPD + RESTART.
;1487    ;       MATCHC REWRITTEN FOR NEW SRM.
;1488    ;       MOVC3 MOVES SOURCE BACKWARDS BY BYTES TO SAVE SPACE.
;1489    ;45     YET ANOTHER VERSION OF PROBE, WHICH IS MORE CAREFUL ABOUT HOW
;1490    ;       ACCESS VIOLATION GETS SET IN REFILL CODE.
;1491    ;       FIX SELPR ROUTINE TO NEVER RETURN 3F, AND TO CORRECTLY CHECK FOR
;1492    ;       REGISTER NUMBER OF THE FORM ***11*, AND IN WPR.5 TO SAVE IPL BITS,
;1493    ;       NOT CLEAR THEM.
;1494    ;       FIX ADDF TO CHECK PRECISELY OF OPRAND APPROX 0 W.R.T. THE OTHER.
;1495    ;       FIX INCREMENTING PC TWICE FOR FLOATING ARITHMETIC.
;1496    ;44     FIX PROBE TO TEST BASE AND BASE+LENGTH-1, NOT BASE+LENGTH.
;1497    ;       FIX PROBE TO RETURN Z=0 IF FINAL PTE NOT VALID BUT ACCESSABLE,
;1498    ;       TO FAULT IF SYSTEM PTE NOT VALID PREVENTING FETCH OF FINAL PTE,
;1499    ;       AND TO SET Z=1 ONLY ON ACCESS VIOLATIONS (INCLUDING LENGTH VIOLATION).
;1500    ;       RECODE SELPR SUBROUTINE TO SAVE 20-ODD LOCATIONS IN MT/FPR.
;1501    ;       ADJUST CHECKS IN MTPR FOR IPL AND ASTLVL TO MEET SOFTWARE NEEDS.
;1502    ;       NEW VERSION OF ERROR HANDLING CODE.  FIXES KNOWN PROBLEMS ***NEW
;1503    ;       STACK MAP ****,  UPC ON TRAPS NOW LOGGED OUT
;1504    ;       FIX JSR IN CM TO WRITE TO R(SRC) AFTER WRITE TO MEMORY.
;1505    ;       FIX CHM TO RESET SP IF FAULT, CHANGE INTERRUPT HANDLING TO ALLOW
;1506    ;       CONDITIONAL ENTRY ON INTERNAL/EXTERNAL INTERRUPT.
;1507    ;       FIX MULF TO WRITE ZERO CORRECTLY,  AND INTEGER MUL (FROM C.FORK)
;1508    ;       TO TAKE CARE OF 2-OP/3-OP.
;1509    ;43     FIX PROBE BY INCLUDING CODE FOR FAULT CASES, WHICH NEVER GOT ENTERED
;1510    ;       FIX AOB, SOB, AND ACB BY CREATING NEW SUBROUTINE IB.TBR WHICH SAVES
;1511    ;       AND RESTORES VA BEFORE INVOKING IB.TBM, WHEN IBUFFER GETS A TB MISS.
;1512    ;       REMOVE TWO STATES FROM ASHL BECAUSE THEY WERE DUPLICATES (ASHL.3, &.5)
;1513    ;       FIX LDPCTX TO SETUP PROCESSOR REGISTER NUMBER OF P1BR BEFORE LOADING.
;1514    ;42     RECODE ASHL TO SET V CORRECTLY ON SHIFT OF 31 PLACES LEFT, AND TO
;1515    ;       SAVE A STATE IN THE BARGAIN.
;1516    ;       FIX FFS/FFC NOT TO DEPEND ON LA WHILE DOING WRITE.DEST
;1517    ;       FIX MTPR TO LOAD PRN INTO STATE WHILE WRITING LENGTH REGISTERS,
;1518    ;       TO ENABLE DISTINGUISHING P1LR AS NEEDED.
;1519    ;       FIX MTPR AND LDPCTX TO ACCEPT 2**21 AS LENGTH OF SEGMENT.
;1520    ;       FIX REMQUE TO CALL ASPC, RATHER THAN MERELY JUMPING THERE.
;1521    ;       FIX SPEC SUBROUTINE FOR (PC)+ LITERALS TO RETURN IMMEDIATELY, RATHER
;1522    ;       THAN GOING TO C.M AND MAKING A WILD MEMORY REFERENCE.
;1523    ;       ISNV WILL DO ERR HALT, AND KSNV WILL RAISE IPL TO 1F CORRECTLY.
;1524    ;       FIX EDIV TO HAVE CORRECT REG # FROM RLOG.
;1525    ;41     FIX LDPCTX AND SVPCTX TO EXPECT RETURN20 FROM KERNEL CHECK.  FIX COUNTER
;1526    ;       IN LDPCTX WHICH TERMINATES LOADING OF STACK POINTERS TO START
;1527    ;       LOADING AT ZERO, AND TERMINATE AT 3.
;1528    ;       FIX ASPC ROUTINE TO HANDLE REGISTER WRITE OPERANDS IN
;1529    ;       WRITE-TYPE SPECIFIERS (AS USED BY REMQUE).
;1530    ;40     MODIFY SELPR ROUTINE IN MTPR/MFPR TO RETURN2 IF SELECTED REGISTER IS
;1531    ;       NOT CURRENT STACK POINTER, RETURN3 IF IT IS CURRENT STACK POINTER.
;1532    ;       THIS MAKES RPR AND WPR ROUTINES USEABLE BY CONSOLE CODE.
;1533    ;       FIX CONSTRAINED BLOCK IN EMUL.
;1534    ;       FIX MOV R,(R)+ IN CM TO INCREMENT PC BY 2 AFTER FLUSH.
;1535    ;39     FIX CASE INSTRUCTIONS TO CALCULATE TMP=SELECTOR-BASE, NOT
```

```
;1536    ;         TMP=BASE-SELECTOR.
;1537    ;38       FIX SOB WITH MEMORY OPERAND TO CLOCK CONDITION CODES FROM LONG
;1538    ;         RESULT, EVEN THOUGH INST.DEP DATA TYPE IS BYTE FOR GETTING BDEST.
;1539    ;         ADD MULF/D, DIVF/D, * - REGISTER MODE ENTRIES.
;1540    ;37       FIX POPR INSTRUCTION TO PROBE (SP) + 4*N -1 INSTEAD OF (SP) + 4*N
;1541    ;         AND TO PRESERVE VA DURING THE PROBE SEQUENCE.
;1542    ;36       PREVENT IB READS WHILE TURNING ON MME OR WRITING TRANSLATION BUFFER.
;1543    ;         ADD MUL2/3, DIV2/3, EMUL, EDIV INSTRUCTIONS.
;1544    ;35       FIX EXTV-TYPE INSTRUCTIONS, WHICH WERE USING D_Q.RIGHT2.SI/ASHR.
;1545    ;         SI/ASHR INHIBITS SHIFTING RIGHT TWO PLACES IN SHF.  ARRRGH...
;1546    ;         FIX SELP.3 IN MOVPR, TO PROCESS PSL.MODE BRANCH CORRECTLY.  IS=1
;1547    ;         GIVES UPC1=0...
;1548    ;34       FIX RETURN ADDRESSES FROM CALL BY ACB TO SPEC, WHICH RETURNS 10/12,
;1549    ;         NOT 60/61.
;1550    ;33       ADD PROBE, MOVD, MNEGD, TSTF/D INSTRUCTIONS.
;1551    ;         VECTOR CODE [1:0] (IN I&E) EQUAL TO 3 WILL HALT WITH VECTOR ID IN PC,
;1552    ;         AND PARAMETERS PUSHED IN THE APPROPRIATE STACK.
;1553    ;         CLEAR TP BIT OF PSL FOR EXCEPTIONS WITHOUT PARAMETERS.
;1554    ;32       ADD INSQUE AND REMQUE INSTRUCTIONS
;1555    ;         FIX BPT & RESERVED ADDRESSING MODE FAULTS TO FAULT INSTEAD OF HANG.
;1556    ;         CLEAR HIGHEST IPR BIT IN SISR IN INTERRUPTS.
;1557    ;31       ADD PROBE CODE TO CALL, PUSHR, ETC.  ADD SPECIAL CASE INSV ON REGISTER.
;1558    ;         MOVE ENTRY POINTS FOR MANY INSTRUCTION TO MEET VARIOUS CONSTRAINTS
;1559    ;         INSTRUCTIONS AFFECTED: MULx2, DIVx2, MOVF, MNEGF, ROTL, INSV, CASEL.
;1560    ;         INCLUDE DEFINITIONS IN DEFNEW, MACRO, AND DOC FOR NEW BRANCHES ON
;1561    ;         Z, C31, AND SRC.PC.
;1562    ;         ADD DOUBLE FLOATING POINT ARITHMETIC INTO STANDARD ASSEMBLY.
;1563    ;30       INCLUDE STRING INSTRUCTIONS IN STANDARD ASSEMBLY.
;1564    ;         CHANGE IRD11 FROM FLOATING LOCATION TO 762 (HEX), AND FIX MSC IN CM.
;1565    ;29       FIX MTPR ,#P1BR TO GET OFFSETS RIGHT.  2**23 MUST BE ADDED TO TEST FOR
;1566    ;         SYSTEM SPACE, BUT MUST BE SUBTRACTED BEFORE STORING VALUE.
;1567    ;         ALSO RE-ARRANGE PROCESSOR REGISTER NUMBERS ACCORDING TO NEW SPEC.
;1568    ;         CHANGE IRD11 FROM 700 (HEX) LOCATION TO FLOATING LOCATION.
;1569    ;         CHANGE MTP, FIX ASH, DIV IN CM.
;1570    ;28       ADD ALU_D FOR ROR, FIX JSR IN COMP MODE(CM).
;1571    ;27       NEATNESS CHANGES TO KEEP LISTING FROM OVERFLOWING PAGES, MINOR UPDATES
;1572    ;         TO DOC.MIC TO KEEP IT CURRENT.
;1573    ;         FIX DM27, ROR, ASR IN COMP MODE.
;1574    ;26       ADD MACROS NECESSARY TO ASSEMBLE CONSOLE CODE INTO STANDARD
;1575    ;         MACRO FILE.
;1576    ;         FIX REI TEST OF PSL<IS> BIT WITH BEN/PSL MODE.  1 IN UPC 1 MEANS IS=0.
;1577    ;         FIX THE ID ADDRESS OF KSP FROM 30 TO 28(HEX).
;1578    ;25       USE MSC FIELD FUNC "CLR.FPD" TO CLR PSL<FPD> IN INTERRUPT HANDLING.
;1579    ;         MSC FIXES TO COMP MODE CODES.
;1580    ;24       CHANGE TO COMP MODE FOR PC UPDATES AND ODD ADDRESS CHECKS.
;1581    ;         SET VA TO 0 WHEN GET ISR, SET CONST 26. IN FE, AND CONST BLOCK
;1582    ;         FIX IN INTERRUPT CODES.
;1583    ;23       PC IS TO BE DECREMENTED BY 1 FOR VAX MODE WHEN PUSHING INTO STACK,
;1584    ;         ALSO FIX BEN ON PSL <IS> BIT, IN INTERRUPT SERVICE.
;1585    ;22       SEVERAL FIXES TO INTERRUPT SERVICE
;1586    ;21       ADD ERROR HANDLING MICRO-CODE TO STAR
;1587    ;         MISC FIXES TO COMPATIBILITY MODE
;1588    ;20       CHANGE USES OF BEN/REI SO UPC 0 IS 'MODE.LSS.ASTLVL"
;1589    ;         CHANGE INITIALIZATION TO STOP IB, AND TO CLEAR PC.
;1590    ;19       ADD COMPATABILITY MODE, MULTIPLY, AND FLOATING POINT.  FIX BFORK
```

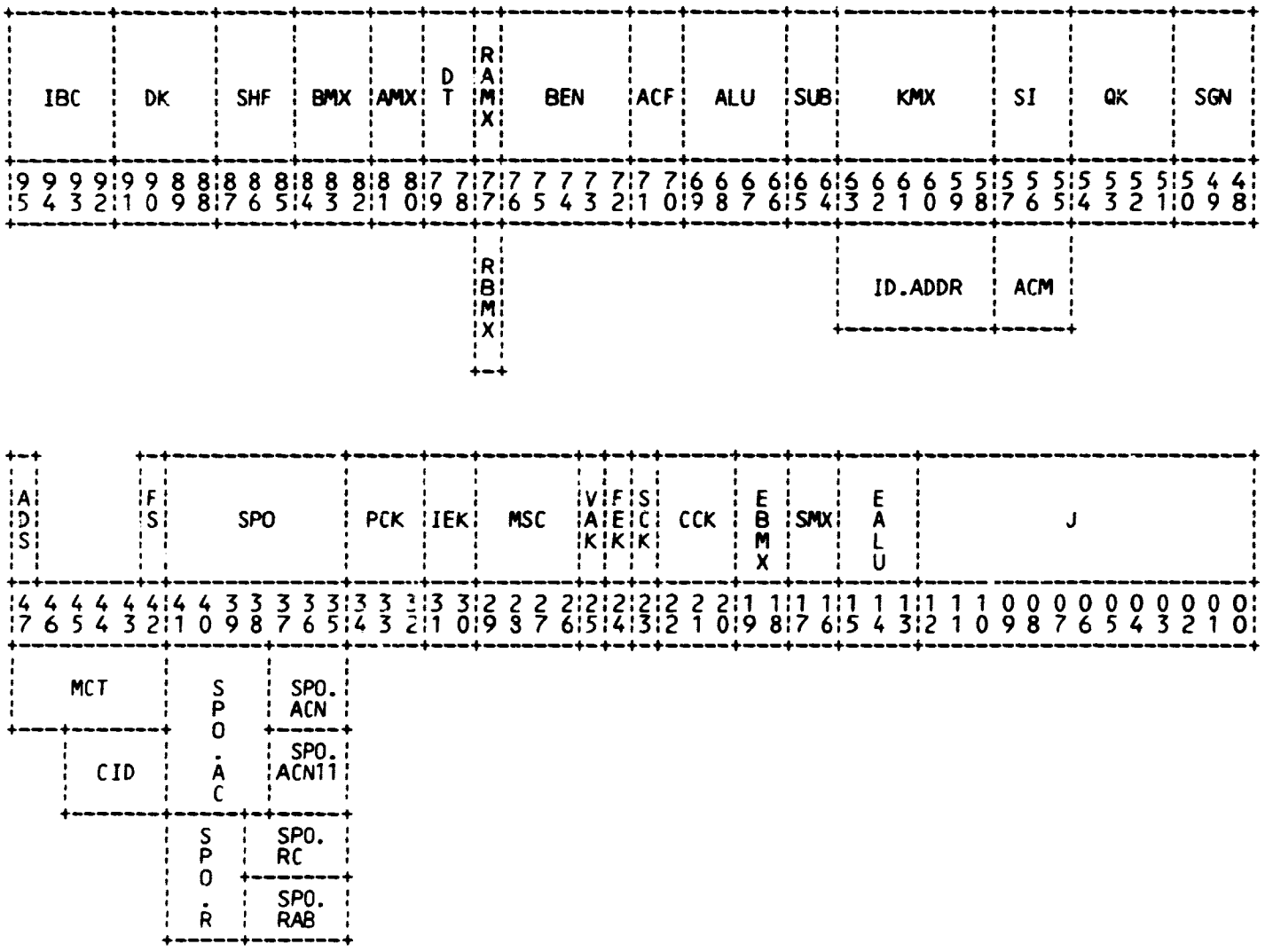
```
;1591    ;       AND CFORK WHEN STORING REGISTER TO CONTROL DATA TYPE FROM INST.DEP.
;1592    ;18     CHANGE ALL BRANCH INSTRUCTIONS (EXCEPT THOSE WHOSE BDEST IS FIRST)
;1593    ;       TO USE IBC/BDEST TO OBTAIN BRANCH DISPLACEMENT.  THIS SAVES IBUFFER
;1594    ;       LOGIC, AND SOLVES A PROBLEM IN BB AND AOB WHICH WAS CAUSING
;1595    ;       THEM TO STALL AT EXECUTION POINT 2, EVEN THOUGH GOING TO EXECUTE.
;1596    ;17     FIX ASHL TO SET N AND Z ON LEFT SHIFTS.  FIX REI TO TEST EALU N
;1597    ;       AFTER COMPARING CURRENT MODE IN OLD PSL WITH THAT IN NEW PSL.
;1598    ;       FIX SOB ON A REGISTER TO AVOID EXCESSIVE PC UPDATES.
;1599    ;       FIX AOB, SOB, BB, BLB, ACB TO USE IBC/CLR.1-5.COND FOR READING
;1600    ;       BRANCH DISPLACEMENTS INSTEAD OF IBC/CLR.1
;1601    ;16     FIX CALL AND RETURN TO INCREMENT PC WHILE DOING LOAD.IB.
;1602    ;       FIX RET TO LOAD CORRECT MASK BITS INTO D FOR TESTING TO
;1603    ;       RESTORE STACK UN-ALIGNMENT.
;1604    ;15     FIXES CVTLW, LB, WB TO ASSIGN ADDRESS 286 TO
;1605    ;       CVT.2.  ALSO FIXES SEVERAL PROBLEMS IN CALL.
;1606    ;14     FIXES TO REI TO USE CORRECT ID BUS ADDRESS FOR STACK POINTERS,
;1607    ;       TO NOTICE COMPATABILITY MODE AND GO TO THE CORRECT IRD STATE
;1608    ;       DEPENDING ON IT, AND TO TEST THE IPL FIELD OF THE NEW PSL, NOT THE
;1609    ;       OLD ONE WHEN THE NEW IS BIT =1.
;1610    ;  ****************************************************************
;1611    ;  *                                                              *
;1612    ;  *    NOTE: THIS IS THE FIRST MICROCODE VERSION ASSEMBLED       *
;1613    ;  *    BY VERSION 30 OF THE MICROASSEMBLER.  IF A DISASTER HAS   *
;1614    ;  *    BEFALLEN IT AND THE ASSEMBLER HAS SCREWED UP, FALL BACK   *
;1615    ;  *    TO VERSION 24 OF THE ASSEMBLER, WHICH IS IN LIB:MICR24    *
;1616    ;  *                                                              *
;1617    ;  ****************************************************************
;1618    ;13     FIX CONDITION CODE SETTING FOR QUAD RESULTS ON BFORK, SO THAT
;1619    ;       MOVQ AND CLRQ DO NOT CHANGE PSL.C
;1620    ;       MOVE THE RETURN ADDRESS ON CFORK IN INDEX MODE EVALUATION TO
;1621    ;       BE 36C, 36D TO CORRESPOND WITH ASPC.
;1622    ;       CHANGE MSC/ FIELD ASSIGNMENTS TO MOVE IRD, INCLUDE SET & CLR FPD.
;1623    ;       CHANGE SPO.ACN/ FIELD TO INCLUDE SP1+1, NOW AVAILABLE, AND CHANGE
;1624    ;       SGN/ FIELD TO MAKE FUNCTION 6 BE OPCODE DEPENDENT.
;1625    ;12     FIX MOVQ TO STORE DESTINATION IN THE RIGHT ORDER.
;1626    ;       REDEFINE RETURNS FROM RPR & WPR AS 10/18 RATHER THAN 17/1F
;1627    ;       TO MINIMIZE WASTAGE OF CONTROL STORE.
;1628    ;11     ADD LDPCTX & SVPCTX.  FIX BUG IN HALT CODE TO JUMP AROUND HALT.INST
;1629    ;       FIX SPECIFIER EVALUATION (4 TIMES) TO CLR SPEC IN @(PC)+ FLOWS.
;1630    ;10     ADDS MEMORY MANAGEMENT AND IBUFF FILL ON TB-MISS ROUTINES
;1631    ;09     FIX BUG IN PUSHR WHICH WAS STORING WRONG SP ON RETURN FROM MPUSH.
;1632    ;       FIX HALT INSTRUCTION TO CHECK FOR KERNEL MODE.
;1633    ;       FIX MTPR TO UPDATE PC AT END.
;1634    ;08     ADD PUSHR, POPR, AND MT/FPR FOR MME, TBIA, TBIS.  FIX SOB & AOB TO
;1635    ;       HOLD ONTO CONTENTS OF IB BYTE 0 UNTIL IRD TESTED.  ADD TESTS IN
;1636    ;       MTPR, MFPR TO VERIFY KERNEL MODE BEFORE MAKING REFERENCE.  FIX
;1637    ;       B.FORK AND WRITE.DEST MACROS TO ALLOW IB READ TO DEFAULT MCT FIELD.
;1638    ;07     INVERT USE OF PTE.VALID BIT IN BEN/1D.  1=NOT VALID
;1639    ;       ADD INTERRUPT/EXCEPTION CODE.
;1640    ;06     MOVED R-R OPTIMIZATION ADDRESSES ON AFORK AND BFORK TO CORESPOND TO
;1641    ;       IBUF.  SEPARATED PC UPDATE FROM CLR.IBXXX MACROS TO SOLVE PROBLEM OF
;1642    ;       DOUBLE UPDATE OF PC IN SEVERAL SITUATIONS.  RECODED AOB TO USE ALU/A-B
;1643    ;       RATHER THAN INST.DEP, BECAUSE A-B GETS OVERFLOW CHECK ON ALU.N BIT.
;1644    ;05     FIXED BEN'S ON IB.TEST TO FORCE 1 IN UPC<02>
;1645    ;       ADDED DT/INST.DEP TO DEFINITION OF MACRO SET.CC(INST),
```

```
;1646    ;       AND RECODED INSTANCES WHERE THIS CAUSES CONFLICT
;1647    ;04     FIXED DEPOSIT GEN REG ROUTINE TO GET DATA FROM ID[T2], NOT
;1648    ;       T1.  ALSO FIXED BFORK AND CFORK USES OF DATA.TYPE? BEN TO EXPECT
;1649    ;       UPC<02> TO BE ASSERTED ON READ/MODIFY, RATHER THAN WRITE.
;1650    ;03     ADDED MICROCODE FOR HALT REQUEST FROM CONSOLE AND HALT INSTRUCTION
;1651    ;       AND ADDED MICROCODE FOR MFPR, MTPR INSTRUCTIONS
;1652    ;02     FIXED UDDT LOAD IBA FUNCTION, WHICH CLOBBERED PC UNWITTINGLY ON IB.FLUSH
;1653    ;01     ADDED CONSOLE AND MICRODEBUGGER MICROCODE
;1654    ;00     BASE VERSION FOR BREADBOARD DEBUG
;1655

                                        ;1656   .BIN
                                        ;1657   .LIST          ;Re-enable full listing
```

```
                                        ;1658   .TOC    "DEFIN.MIC"
                                        ;1659   .TOC    "Revision 1.1"
                                        ;1660   ;        P. R. Guilbault
                                        ;1661
;1662   .NOBIN                    ;DON'T LEAVE LISTING SPACE FOR BINARY
;1663   .RTOL                     ;BITS NUMBERED FROM 0 ON THE RIGHT OF A FIELD
;1664   .HEXADECIMAL              ;RADIX 16
;1665   .NOCREF                   ;SUPPRESS CREF ON UNINTERESTING FIELDS
;1666   .SET/NATIVE=1             ;Validity check for CCK micro orders
;1667
;1668   .FILLERPAGE
;1669   .BOUNDS/PCS:0000,0FFF
;1670   .BOUNDS/WCS:1000,17FF
;1671   .BOUNDS/G&H:1800,1BFF
;1672
;1673   .TOC    ""      Revision History"
;1674
;1675   ; 01    Add .BOUNDS pseudo-op to get report by file.
;1676   ;       Clean up CCK definitions and give more info about micro-orders
;1677   ; 00    State of history
;1678
;1679   .NOLIST           ;Disable listing of PCS code for quickie assemblies
```

```
:1680   .TOC     ""      Machine definition    : Control word chart"
:1681
:1682
:1683   ;
:1684   ;
:1685   ;
:1686   ;
:1687   ;
:1688   ;
:1689   ;
:1690   ;
:1691   ;
:1692   ;
:1693   ;
:1694   ;
:1695   ;
:1696   ;
:1697   ;
:1698   ;
:1699   ;
:1700   ;
:1701   ;
:1702   ;
:1703   ;
:1704   ;
:1705   ;
:1706   ;
:1707   ;
:1708   ;
:1709   ;
:1710   ;
:1711   ;
:1712   ;
:1713   ;
:1714   ;
:1715   ;
:1716   ;
:1717   ;
:1718   ;
:1719   ;
:1720   ;
:1721   ;
:1722   ;
:1723   ;
:1724   ;
:1725   ;
:1726   ;
:1727   ;
```



```
;1728   .TOC    "       Machine definition    : ACF, ACM, ADS, ALU, AMX"
;1729
;1730   ACF/=<71:70>,.DEFAULT=0          ;ACCELERATOR CONTROL
;1731           NOP=0
;1732           SYNC=1
;1733           TRAP=2
;1734           CONTROL=3                ;ACCELLERATOR-DEPENDENT CONTROL FUNCTION
;1735
;1736   ACM/=<57:55>                     ;ACCELERATOR MISCELLANEOUS CONTROL
;1737           PWR.UP=0
;1738           ABORT=1                  ;RETURN ACCEL TO MONITORING IRD
;1739           POLY.DONE=6
;1740
;1741   ADS/=<47:47>                     ;ADDRESS SELECT
;1742           VA=0
;1743           IBA=1
;1744
;1745   ALU/=<69:66>,.DEFAULT=0F         ;ALU CONTROL FUNCTIONS
;1746           A-B=00
;1747           A-B.RLOG=01
;1748           A-B-1=02
;1749           INST.DEP=03                      ;INSTRUCTION DEPENDENT
;1750           A+B+1=04
;1751           A+B=05
;1752           A+B.RLOG=06
;1753           ORNOT=07                         ;A .OR. .NOT
;1754           XOR=08                           ;A .XOR. B
;1755           ANDNOT=09                        ;A .AND. .NOT. B
;1756           NOTA=0A                          ;.NOT. A
;1757           A+B+PSL.C=0B
;1758           OR=0C                            ;A .OR. B
;1759           AND=0D                           ;A .AND. B
;1760           B=0E
;1761           A=0F
;1762
;1763   AMX/=<81:80>                     ;AMX TO ALU
;1764           LA=0
;1765           RAMX=1
;1766           RAMX.SXT=2               ;RAMX SIGN EXTENDED ACCORDING TO DT
;1767           RAMX.0XT=3               ;RAMX ZERO EXTENDED. 0XT(L)=0
```

```
;1768    .TOC    "       Machine definition    : BEN, BMX"
;1769
;1770    BEN/=<76:72>,.DEFAULT=0          ;BRANCH ENABLE
;1771            NOP=0                    ;NO BRANCH
;1772            Z=1                      ; ALU Z
;1773            ROR=2                    ;LA<1>, PSL<C>, LA<0>
;1774            C31=3                    ; ALU C31, 0
;1775            IRC.ROM=4                ;OUTPUT OF EXTENDED IRC-ROM
;1776            IB.0=5                   ;IB 0 READY ?
;1777            ACCEL=6                  ;CODE FROM ACCELERATOR
;1778            DATA.TYPE=8              ;(VAX MODE) *, ASRC+VSRC, ASRC+Q+D
;1779                                     ;   0--NORMAL, 1--QUAD OR DOUBLE
;1780                                     ;   2--FIELD,  3--ADDRESS
;1781            END.DP1=8                ;(-11 MODE) *, 0 CLASS, J CLASS+DM27
;1782            IR2-1=9                  ;(VAX MODE) *, IR<2:1>
;1783            PC.MODES=9               ;(-11 MODE) *, SM47+SM57+DM47+DM57, DST R=PC
;1784            REI=0A                   ;(VAX MODE) MODE.LSS.ASTLVL, *, *
;1785            SRC.PC=0A                ;(-11 MODE) SRC R=PC
;1786            IB.TEST=0B               ; 0--TB MISS, 1--ERROR
;1787                                     ; 2--STALL, 3--DATA OK
;1788            MUL=0C                   ;SC.NE.0, D<1:0>
;1789            SIGNS=0D                 ;Q<31>, D.NE.0, D<31>
;1790            INTERRUPT=0E             ;AC LOW, INTERNAL INTERRUPT, INT REQ
;1791            DECIMAL=0F               ;0, D BYTE 0 VALID DIGIT, D2-0 NEG SIGN
;1792    ;       UTRAP=10                 ;MICROTRAP DISPATCH VECTOR
;1793            LAST.REF=11              ;-FPD, NESTED ERROR, LOW TWO BITS:
;1794                                     ; 0--READ INTERLOCK, 1--READ, READ CHK
;1795                                     ; 2--WRITE, 3--READ, WRITE CHK
;1796            EALU=12                  ;EALU N, EALU Z, SC.NEQ.0, SS
;1797            SC=14                    ;SC<9:8>.NE.0, SC.GT.0, SC<9:5>.NE.0
;1798            ALU1-0=15                ;RLOG EMPTY, ALU<1:0>=0, ALU<1>, ALU<0>
;1799                                     ; (ALU BITS FROM PREVIOUS STATE)
;1800            STATE7-4=16              ;STATE <7:4>
;1801            STATE3-0=17              ;STATE <3:0>
;1802            D.BYTES=18               ;BYTES 3, 2, 1, 0 OF D.NE.0
;1803            D3-0=19                  ;D<3:0>
;1804            PSL.CC=1A                ;N,Z,V,C OF PSL
;1805            ALU=1B                   ;ALU N, ALU Z, IR<0>, ALU C31
;1806            PSL.MODE=1C              ;-VA<31:30>, -CONSOLE, IS+CM, KERNEL
;1807            TB.TEST=1D               ;PTE VALID, ALIGNED, QUAD, +
;1808                                     ; 0--TRANSLATION OK, 1--WR CHK AND M=0
;1809                                     ; 2--ACCESS VIOLATION, 3--TB MISS
;1810
;1811    BMX/=<84:82>                     ;BMX TO ALU
;1812            MASK=0                   ;A 0 IN THE BIT SELECTED BY SC<4:0>
;1813            PC.OR.LB=1               ;LB UNLESS R=PC, THEN PC
;1814            PACKED.FL=2              ;PACKED FLOATING
;1815            LB=3
;1816            LC=4
;1817            PC=5
;1818            KMX=6
;1819            RBMX=7                   ;D OR Q
```

```
;1820    .TOC     ""       Machine definition    . CCK, CID, DK, DT""
;1821
;1822    CCK/=<22:20>,.DEFAULT=0 ;CONDITION CODES
;1823                                                    ;Note : * = RESERVED OPERATION, ''ALU SIGN'' AND ''AMX SIGN'' ARE SIZE DEPENDENT
;1824                                                    ;+-----------------------------------------+-----------------------------------------+
;1825                                                    ;|          NATIVE MODE PSL                |          COMPATIBILITY MODE PSL         |
;1826                                                    ;+-----------+-----------------+---+---+-----------+-----------------+---+---+
;1827                                                    ;|     N     |        Z        | V | C |     N     |        Z        | V | C |
;1828                                                    ;+-----------+-----------------+---+---+-----------+-----------------+---+---+
;1829            NOP=0                                  ;|     N     |        Z        | V | C |     N     |        Z        | V | C |
;1830            LOAD.UBCC=1                            ;|     N     |        Z        | V | C |     N     |        Z        | V | C |
;1831            SET.V=2                 ,.VALIDITY=<V1>;|     N     |        Z        | 1 | C |     *     |        *        | * | * |
;1832            N_AMX.Z_TST.VC_VC=3                    ;| AMX SIGN  | Z.and.(ALU.eq.0)| V | C | AMX SIGN  | Z.and.(ALU.eq.0)| V | C |
;1833            ROR=4                   ,.VALIDITY=<V0>;|     *     |        *        | * | * | ALU SIGN  |     ALU.eq.0    | 0 | AMX<0> |
;1834            NZ_ALU.VC_0=5                          ;| ALU SIGN  |     ALU.eq.0    | 0 | 0 | ALU SIGN  |     ALU.eq.0    | 0 |   0    |
;1835            NZ_ALU.VC_VC=6          ,.VALIDITY=<V1>;| ALU SIGN  |     ALU.eq.0    | V | C |           |                 |   |        |
;1836            C_AMX0=6                ,.VALIDITY=<V0>;|           |                 |   |   |     N     |        Z        | V | AMX<0> |
;1837            INST.DEP=7                             ;|                                         Instruction dependent                         |
;1838                                                    ;+-----------+-----------------+---+---+-----------+-----------------+---+---+
;1839
;1840    CID/=<45:42>                    ;CONSOLE & ID BUS CONTROL IF FS/1
;1841            NOP=1                   ;DEFAULT, ALLOW AUTO IB READ
;1842            ACK=5                   ;SET CONSOLE ACKNOWLEGE FLAG
;1843            CONT=7                  ;CLEAR CONSOLE MODE
;1844            READ.SC=9               ;READ ID BUS REG SELECTED BY SC
;1845            READ.KMX=0B             ;READ ID BUS REG SELECTED BY UKMX
;1846            WRITE.SC=0D             ;WRITE REG SELECTED BY SC
;1847            WRITE.KMX=0F            ;WRITE REG SELECTED BY UKMX
;1848
;1849    DK/=<91:88>,.DEFAULT=0
;1850            NOP=0                   ;DEFAULT, HOLD
;1851            LEFT2=1                 ;DOUBLE SHIFT LEFT
;1852            RIGHT2=2                ;DOUBLE SHIFT RIGHT
;1853            DIV=4                   ;IF NOT ALU CRY, SHIFT LEFT
;1854                                    ; ELSE LOAD FROM SHF
;1855            LEFT=5                  ;SHIFT LEFT
;1856            RIGHT=6                 ;SHIFT RIGHT
;1857            SHF=8                   ;LOAD SHF MUX, INTEGER FORMAT
;1858            SHF.FL=9                ;LOAD SHF MUX, UNPACKED FLOATING FORMAT
;1859            ACCEL=0A                ;LOAD ACCELERATOR DATA FROM DF BUS
;1860            BYTE.SWAP=0B            ;REFLECT BYTES AROUND BIT 16
;1861            Q=0C                    ;LOAD Q THRU DAL
;1862            DAL.SC=0D               ;LOAD DAL[SC]
;1863            DAL.SV=0E               ;LOAD DAL[SHF VAL]
;1864            CLR=0F                  ;LOAD ZEROS
;1865
;1866    DT/=<79:78>,.DEFAULT=0          ;DATA TYPE
;1867                                    ;CONTROLS AMX SIGN/ZERO EXTENDER, SHF AMOUNT,
;1868                                    ;CONDITION CODE SETTING, AND MEMORY REFERENCES
;1869            LONG=0                  ;DEFAULT
;1870            WORD=1
;1871            BYTE=2
;1872            INST.DEP=3              ;INSTRUCTION DEPENDENT -
;1873                                    ;ANY OF ABOVE, OR QUAD/DOUBLE
```

```
;1874    .TOC    ""      Machine definition    : EALU, EBMX, FEK, FS, IEK, IBC""
;1875
;1876    EALU/=<15:13>                   ;EXPONENT ALU
;1877            A=0
;1878            OR=1
;1879            ANDNOT=2
;1880            B=3
;1881            A+B=4
;1882            A-B=5
;1883            A+1=6
;1884            NABS.A-B=7              ;-ABS(A-B)
;1885
;1886    EBMX/=<19:18>                   ;EBMX TO EALU
;1887            FE=0                    ;DEFAULT
;1888            KMX=1
;1889            AMX.EXP=2
;1890            SHF.VAL=3               ;SHIFT VALUE
;1891
;1892    FEK/=<24:24>,.DEFAULT=0         ;FE REGISTER CONTROL
;1893            NOP=0                   ;DEFAULT, HOLD
;1894            LOAD=1
;1895
;1896    FS/=<42:42>                     ;FUNCTION SELECT FOR 43-46
;1897            MCT=0                   ;ENABLE MEMORY CONTROL
;1898            CID=1                   ;ENABLE ID BUS AND CONSOLE CONTROL
;1899
;1900    IEK/=<31:30>                    ;INTERRUPT AND EXCEPTION ACKNOWLEDGE
;1901            NOP=0
;1902            ISTR=1                  ;STROBE INTERRUPT REQUESTS
;1903            IACK=2                  ;INTERRUPT ACKNOWLEDGE
;1904            EACK=3                  ;EXCEPTION ACKNOWLEDGE
;1905
;1906    IBC/=<95:92>,.DEFAULT=0         ;IBUF CONTROL FUNCTIONS
;1907            NOP=0                   ;DEFAULT
;1908            STOP=1
;1909            FLUSH=2                 ;FLUSH IB AND LOAD IBA
;1910            START=3
;1911            CLR.0.1=4               ;CLEAR BYTES 0,1 (-11 OPCODE)
;1912            CLR.2.3=5               ;CLEAR BYTES 2,3 (-11 ISTREAM DATA)
;1913            BDEST=7                 ;TRANSFER BRANCH DISPLACEMENT
;1914            CLR.0=0C                ;CLEAR BYTE 0 (VAX OPCODE)
;1915            CLR.1=0D                ;CLEAR BYTE 1 (VAX SPECIFIER)
;1916            CLR.0-3=0E              ;CLEAR BYTES 0-3 (-11 OP & DATA)
;1917            CLR.1-5.COND=0F         ;CLEAR BYTES 1-5 CONDITIONALLY
;1918                                    ; IF THERE IS NO SPECIFIER EVALUATION,
;1919                                    ; CLEAR NOTHING.  IF A SELF-CONTAINED
;1920                                    ; SPECIFIER, CLEAR IT.  IF IMMEDIATE,
;1921                                    ; ABSOLUTE, OR DISPLACEMENT, CLEAR THE
;1922                                    ; ISTREAM LITERAL.
```

```
;1923    .TOC    "       Machine definition    : ID.ADDR, J"
;1924
;1925    ID.ADDR/=<63:58>                ;ID BUS ADDRESS
;1926            IBUF=0                  ;RD     ;SPECIFIER/LITERAL DATA FROM IB
;1927            DAY.TIME=1              ;RD+WR  ;CURRENT TIME OF DAY...
;1928                                            ; MUST READ UNTIL STOPS CHANGING
;1929            SYS.ID=3                ;RD     ;SYSTEM IDENTIFICATION
;1930            RXCS=4                  ;RD+WR  ;CONSOLE RECIEVE CONTROL/STATUS
;1931            RXDB=5                  ;RD     ;CONSOLE RECIEVE DATA BUFFER
;1932                                            ; (TO-ID REGISTER)
;1933            TXCS=6                  ;RD+WR  ;CONSOLE TRANSMIT CONTROL/STATUS
;1934            TXDB=7                  ;WR     ;CONSOLE TRANSMIT DATA BUFFER
;1935                                            ; (FROM-ID REGISTER)
;1936    ;       DQ=8                            ;DATA PATH D/Q REGISTERS (MAINT ONLY)
;1937            NXT.PER=9               ;WR     ;INTERVAL CLOCK NEXT PERIOD REGISTER
;1938            CLK.CS=0A               ;RD+WR  ;INTERVAL CLOCK CONTROL/STATUS
;1939            INTERVAL=0B             ;RD     ;CURRENT INTERVAL COUNT
;1940            CES=0C                  ;RD+WR  ;CPU ERROR/STATUS
;1941            VECTOR=0D               ;RD+WR  ;EXCEPTION & INTERRUPT CONTROL
;1942            SIR=0E                  ;RD+WR  ;SOFTWARE INTERRUPT REGISTER
;1943            PSL=0F                  ;RD+WR  ;PROCESSOR STATUS LONGWORD
;1944            TBUF=10                         ;TRANSLATION BUFFER DATA
;1945            TBER0=12                        ;TB ERROR/STATUS 0
;1946            TBER1=13                        ;TB ERROR/STATUS 1
;1947            ACC.0=14                        ;ACCELERATOR REGISTER #0
;1948            ACC.1=15                        ;ACCELERATOR REGISTER #1
;1949            ACC.2=16                        ;ACCELERATOR REGISTER #2
;1950            ACC.CS=17                       ;ACCELERATOR CONTROL/STATUS
;1951            SILO=18                         ;NEXT ITEM FROM SBI HISTORY
;1952            SBI.ERR=19                      ;SBI ERROR REGISTER
;1953            TIME.ADDR=1A                    ;SBI TIMEOUT ADDRESS
;1954            FAULT=1B                        ;FAULT/STATUS
;1955            COMP=1C                         ;SBI SILO COMPARATOR
;1956            MAINT=1D                        ;SBI MAINTENANCE
;1957            PARITY=1E                       ;CACHE PARITY
;1958            USTACK=20                       ;MICROSTACK
;1959            UBREAK=21                       ;MICRO BREAK
;1960            WCS.ADDR=22
;1961            WCS.DATA=23             ;WR     ;WRITING WCS COUNTS ADDRESS
```

```
;1962    ;ID BUS ADDRESSES CONTINUED.  ADDRESSES 24-3F ARE RAM LOCATIONS
;1963
;1964            P0BR=24                         ;PROCESS SPACE 0 BASE REGISTER
;1965            P1BR=25                         ;PROCESS SPACE 1 BASE REGISTER
;1966            SBR=26                          ;SYSTEM SPACE BASE REGISTER
;1967            KSP=28                          ;KERNEL STACK POINTER
;1968            ESP=29                          ;EXEC STACK POINTER
;1969            SSP=2A                          ;SUPERVISOR STACK POINTER
;1970            USP=2B                          ;USER STACK POINTER
;1971            ISP=2C                          ;INTERRUPT STACK POINTER
;1972            FPDA=2D
;1973            D.SV=2E
;1974            Q.SV=2F
;1975            T0=30                           ;GENERAL TEMPS
;1976            T1=31
;1977            T2=32
;1978            T3=33
;1979            T4=34
;1980            T5=35
;1981            T6=36
;1982            T7=37
;1983            T8=38
;1984            T9=39
;1985            PCBB=3A                         ;PROCESS CONTROL BLOCK BASE
;1986            SCBB=3B                         ;SYSTEM CONTROL BLOCK BASE
;1987            P0LR=3C                         ;PROCESS 0 LENGTH REGISTER
;1988            P1LR=3D                         ;PROCESS 1 LENGTH REGISTER
;1989            SLR=3E                          ;SYSTEM LENGTH REGISTER
;1990
;1991    .CREF
;1992
;1993    J/=<12:0>,.NEXTADDRESS                  ;NEXT MICRO WORD ADDRESS
;1994
;1995    .NOCREF
```

```
;1996    .TOC     ''       Machine definition    : KMX''
;1997
;1998    KMX/=<63:58>              ;CONSTANTS OR # FROM FK
;1999            .8=0              ;#8 FROM FK
;2000            .1=1              ;#1 FROM FK
;2001            .2=2              ;#2 FROM FK
;2002            .3=3              ;#3 FROM FK
;2003            .4=4              ;#4 FROM FK
;2004            SP1.CON=5         ;SPECIFIER 1 CONSTANT
;2005            SP2.CON=6         ;SECIFIER 2 CONSTANT (-11 MODE)
;2006            ZERO=6            ; OR ZEROS (VAX MODE)
;2007            SC=7              ;SC[9:0] FROM FK
;2008                              ;8 - 3F: CONSTANTS (1 CYCLE SETUP IF ALU IN ARITH MODE)
;2009                              ;DECIMAL VALUE OF CONSTANT
;2010            .14=8             ;20      (AF,JL,MH)
;2011            .A0=9             ;160     (AF,JL)
;2012            .34=0A            ;52      (AF)
;2013            .28=0B            ;40      (AF)
;2014            .40=0C            ;64      (AF,JL,MH,TF)
;2015            .50=0D            ;80      (AF,MH)
;2016            .7FF0=0E          ;?       (TF) ****MACHINE-DEPENDENT!!!
;2017            .EF=0F            ;239     (JL)
;2018            .80=10            ;128     (AF,JL,MH,TF)
;2019            .8000=11          ;-32768  (AF)
;2020            .FF=12            ;255     (MH,TF)
;2021            .FF00=13          ;-256    (MH,AF,JL)
;2022            .1E=14            ;30      (AF)
;2023            .3F=15            ;63      (MH,AF,TF)
;2024            .7F=16            ;127
;2025            .7=17             ;7       (AF,MH)
;2026            .F=18             ;15      (MH,CM,AF,TF)
;2027            .10=19            ;16      (MH,AF,JL,TF)
;2028            .FFE8=1A          ;-24     (MH,TF)
;2029            .FFF0=1B          ;-16     (CM,JL,TF,MH)
;2030            .FFF8=1C          ;-8      (CM,TF,MH)
;2031            .20=1D            ;32      (CM,JL,MH,TF)
;2032            .30=1E            ;48      (CM,AF,MH,TF)
;2033            .18=1F            ;24      (MH,AF,TF)
;2034            .3FF=20           ;1023    (CM)
;2035            .C=21             ;12      (CM,JL,TF,MH)
;2036            .D=22             ;13      (TF)
;2037            .1F=23            ;31      (AF,JL,MH,TF)
;2038            .1F00=24          ;7936    (JL,MH)
;2039            .B0=25            ;176     (MH)
;2040            .E003=26          ;        (CM)
;2041            .7C=27            ;124     (AF)
;2042            .FFE0=28          ;-32     (JL)
;2043            .60=29            ;96      (TF)
;2044    ;       SPARE=2A
;2045            .DFCF=2B          ;?       (JL)
;2046            .4000=2C          ;?       (TF)****MACHINE-DEPENDENT!!
;2047            .FFF1=2D          ;-15     (AF)****MACHINE-DEPENDENT!!
;2048            .19=2E            ;25      (AF)
;2049            .FFF9=2F          ;-7      (AF)
```

```
;2050    ; KMX DEFINITION CONTINUED
;2051
;2052            .FFFF=30        ;-1     (MH,JL,TF)
;2053            .88=31          ;136    (AF)
;2054            .3030=32        ;?      (TF)
;2055            .F0=33          ;240    (TF)
;2056            .C0=34          ;19.    (TF,MH)
;2057            .6=35           ;6      (CM,JL,TF)
;2058            .9=36           ;9      (CM)
;2059            .FFF6=37        ;-10    (CM)
;2060            .FFF5=38        ;-11    (CM)
;2061            .1A=39          ;26     (CM,AF,TF)
;2062            .24=3A          ;36     (CM,MH)
;2063            .1B=3B          ;27     (CM,AF,TF)
;2064            .FFFC=3C        ;-4     (CM,TF,MH)
;2065            .A=3D           ;10     (AF,MH)
;2066            .7E=3E          ;126    (AF,TF)
;2067    ;       SPARE=3F
```

```
;2068    .TOC    ""      Machine definition    : MCT, MSC""
;2069
;2070    MCT/=<47:42>,.DEFAULT=3E                ;MEMORY CONTROL
;2071            TEST.RCHK=00                    ;TEST TBUF WITH READ CHECK
;2072            MEM.NOP=02                      ;NEITHER CPU NOR IB GETS MEM CYCLE
;2073            TEST.WCHK=04                    ;TEST TBUF WITH WRITE CHECK
;2074            WRITE.V.NOCHK=0A                ;WRITE, INHIBIT TRAPS
;2075            WRITE.V.WCHK=0C                 ;WRITE, NORMAL VARIETY
;2076            LOCKWRITE.V.XCHK=0E             ;INTERLOCK WRITE, VIRTUAL ADDRESS
;2077            READ.V.RCHK=10                  ;READ, NORMAL VARIETY
;2078            READ.V.NOCHK=12                 ;READ, INHIBIT TRAPS
;2079            READ.V.WCHK=14                  ;READ FOR MODIFY
;2080            READ.V.IBCHK=16                 ;READ, CHECK CONTROLLED BY IBUFFER
;2081            READ.V.NEWPC=18                 ;BEGIN NEW INSTRUCTION STREAM
;2082                                            ; DATA GOES TO INSTRUCTION BUFFER
;2083            LOCKREAD.V.NOCHK=1A             ;INTERLOCK READ, INHIBIT CHECK
;2084            LOCKREAD.V.WCHK=1C              ;INTERLOCK READ, NORMAL VARIETY
;2085            SBI.HOLD=20                     ;STOP ALL SBI ACTIVITY
;2086            SBI.HOLD+UNJAM=22               ;RESET SBI
;2087            INVALIDATE=24                   ;CLEAR CACHE ENTRIES
;2088            VALIDATE=26                     ;MICRODIAGNOSTIC FORCE VALID
;2089            EXTWRITE.P=28                   ;EXTENDED WRITE TO CLEAR MOS ERRORS
;2090            WRITE.P=2A                      ;WRITE, PHYSICAL
;2091            LOCKWRITE.P=2E                  ;INTERLOCK WRITE, PHYSICAL
;2092            READ.P=32                       ;READ, PHYSICAL
;2093            READ.INT.SUM=36                 ;INTERRUPT SUMMARY READ
;2094            LOCKREAD.P=3A                   ;INTERLOCK READ, PHYSICAL
;2095            ALLOW.IB.READ=3E                ;GIVE IB A CYCLE IF IT WANTS ONE
;2096
;2097    MSC/=<29:26>,.DEFAULT=0
;2098            NOP=0                           ;DEFAULT
;2099            CHK.CHM=01                      ;CREATE NEW PSL FOR CHM
;2100            CHK.FLT.OPR=02                  ;UTRAP IF ALU<15>=1, ALU<14:7>=0
;2101            CHK.ODD.ADDR=03
;2102            IRD=04                          ;THIS STATE IS INSTRUCTION DECODE
;2103            LOAD.STATE=05
;2104            LOAD.ACC.CC=06                  ;TAKE CONDITION CODES FROM ACCELERATOR
;2105            READ.RLOG=07                    ;(AND POP RLOG STACK)
;2106            CLR.FPD=08                      ;CLEAR PSL<FPD> BIT
;2107            SET.FPD=09                      ;SET SAME
;2108            CLR.NEST.ERR=0A                 ;CLR NESTED ERROR FLAG IN CPU STATUS
;2109            SET.NEST.ERR=0B                 ;SET SAME
;2110            SECOND.REF=0C                   ;OF UNALIGNED DATA REFERENCE
;2111            RETRY.NO.TRAP=0D                ;APPLY SAVED CONTEXT, INHIBIT TRAPS
;2112            RETRY.TRAP=0E                   ;APPLY SAVED CONTEXT TO THIS REF
;2113            INH.CM.ADDR=0F                  ;ALLOW USE OF FULL 32-BIT ADDRESS
```

```
;2114    .TOC    "       Machine definition    : PCK, QK, RAMX, RBMX"
;2115
;2116    PCK/=<34:32>,.DEFAULT=0          ;ADDRESS COUNT CONTROL
;2117            NOP=0                    ;DEFAULT
;2118            PC_VA=1
;2119            PC_IBA=2
;2120            VA+4=3                   ;VA_VA+4
;2121            PC+1=4                   ;PC_PC+1
;2122            PC+2=5                   ;PC_PC+2
;2123            PC+4=6                   ;PC_PC+4
;2124            PC+N=7                   ;PC_PC+N,  N IS DETERMINED BY INSTR BUFFER
;2125
;2126    QK/=<54:51>,.DEFAULT=0
;2127            NOP=0                    ;DEFAULT, HOLD
;2128            LEFT2=1                  ;DOUBLE SHIFT LEFT 2
;2129            RIGHT2=2                 ;DOUBLE SHIFT RIGHT 2
;2130            LEFT=5
;2131            RIGHT=6
;2132            SHF=8                    ;LOAD SHF, INTEGER FORMAT
;2133            SHF.FL=9                 ;LOAD SHF, UNPACKED FLOATING FORMAT
;2134            DEC.CON=0A               ;DECIMAL CONSTANT = 6'S IN EACH NIBBLE
;2135                                     ;FOR WHICH ALU CRY OUT IS FALSE
;2136            ACCEL=0B                 ;LOAD ACCELERATOR DATA FROM DF BUS
;2137            D=0C
;2138            ID=0E                    ;LOAD ID BUS
;2139            CLR=0F                   ;LOAD ZERO
;2140
;2141    RAMX/=<77:77>,.DEFAULT=0         ;DATA PATH MIXER TO AMX
;2142            D=0                      ;DEFAULT
;2143            C=1
;2144
;2145    RBMX/=<77:77>                    ;DATA PATH MIXER TO BMX.  SAME BIT AS RAMX
;2146            Q=0
;2147            D=1
```

```
;2148   .TOC     ""      Machine definition    : SCK, SGN, SHF, SI, SMX"
;2149
;2150   SCK/=<23:23>,.DEFAULT=0          ;SC REGISTER CONTROL
;2151           NOP=0                   ;DEFAULT, HOLD
;2152           LOAD=1                  ;LOAD SMX<09:00>
;2153
;2154   SGN/=<50:48>,.DEFAULT=0          ;SIGN CONTROLS
;2155           NOP=0                   ;DEFAULT
;2156           LOAD.SS=1               ;SS_ALU<15>
;2157           SS.FROM.SD=2            ;SS_SD
;2158           NOT.SD=3                ;SD_NOT SD
;2159           SD.FROM.SS=4            ;SD_SS
;2160           SS.XOR.ALU=5            ;SD_ALU<15>, SS_SS.XOR.ALU<15>
;2161           ADD.SUB=6               ;SD_ALU<15>, SS_SS.XOR.ALU<15>.XOR.IR<1>
;2162           CLR.SD+SS=7             ;CLEAR BOTH
;2163
;2164   SHF/=<87:85>,.DEFAULT=0          ;ALU SHIFTER CONTROLS
;2165           ALU=0                   ;DEFAULT, SHF_ALU
;2166           LEFT=1                  ;SHF_ALU(L1), INSERT SI CNTL
;2167           RIGHT=2                 ;SHF_ALU(R1), INSERT SI CNTL
;2168           ALU.DT=3                ;SHF_ALU(DT: L0,L1,L2,L3), INSERT 0
;2169           RIGHT2=4                ;SHF_ALU(R2), INSERT SI CNTL
;2170           LEFT3=5                 ;SHF_ALU(L3)
;2171
;2172   SI/=<57:55>,.DEFAULT=3           ;SHIFT INPUT CONTROLS
;2173                                   ;       SHF     D       Q
;2174                                   ;       ---     -       -
;2175           DIVD=0                  ;       PSL<N>  Q31     ALU C31
;2176           ASHR=1                  ;       ALU 31  Q0      Q31
;2177           ASHL=2                  ;       0       0       D31
;2178           ZERO=3                  ;       0       0       0
;2179   ;       SPARE=4
;2180           DIV=5                   ;       Q31     Q31     ALU C31
;2181           MUL+=6                  ;       0       ALU 0,1 0
;2182           MUL-=7                  ;       1       ALU 0,1 1
;2183
;2184   SMX/=<17:16>                    ;MIXER TO SC
;2185           EALU=0                  ;EALU <9:0>
;2186           FE=1                    ;FE<9:0>
;2187           ALU=2                   ;ALU<09:00>
;2188           ALU.EXP=3               ;ALU<14:07>
```

```
;2189    .TOC     ''       Machine definition    : SPO, SPO.AC, SPO.ACN, SPO.ACN11, SPO.R''
;2190
;2191    SPO/=<41:35>,.DEFAULT=0                  ;SCRATCH PAD OPCODE, 7 BITS
;2192            NOP=0                            ;DEFAULT
;2193            LOAD.LC.SC=6                     ;LOAD LC, ADR=SC[03:00]
;2194            WRITE.RC.SC=7                    ;WRITE RC, ADR=SC[03:00]
;2195
;2196    SPO.AC/=<41:38>                          ;4 FUNCTION BITS OF SPO FIELD
;2197            LOAD.LAB=1                       ;LOAD LA, LB FROM R(ACN)
;2198            LOAD.LA=2                        ;LOAD LA RN, HOLD LB
;2199            WRITE.RAB=3                      ;WRITE RA, RB (ACN)
;2200
;2201    SPO.ACN/=<37:35>                 ;AC NUMBER IN SPO FIELD
;2202                                     ;VAX MODE       RA              RB
;2203            SP1.SP1=0                ;0              SP1 R           SP1 R
;2204            SP2.SP2=1                ;1              SP2 R           SP2 R
;2205            SP2.SP1=2                ;2              SP2 R           SP1 R
;2206            PRN=3                    ;3              PRN             PRN
;2207            PRN+1=4                  ;4              PRN+1           PRN+1
;2208            SC=5                     ;5              SC<03:00>       SC<03:00>
;2209            SP1+1=6                  ;6      SP1 R+1         SP1 R+1
;2210
;2211    SPO.ACN11/=<37:35>               ;AC NUMBER IN SPO FIELD -- 11 MODE
;2212                                     ;-11 MODE         RA              RB
;2213            SRC.SRC=0                ;   0           SRC R           SRC R
;2214            DST.DST=1                ;   1           DST R           DST R
;2215            DST.SRC=2                ;   2           DST R           SRC R
;2216    ;       SRC.SRC=3                ;   3           SRC R           SRC R
;2217            SRC.OR.1=4               ;   4           SRC R .OR. 1    SRC R .OR. 1
;2218            SC=5                     ;   5           SC<03:00>       SC<03:00>
;2219
;2220    SPO.R/=<41:39>                   ;SCRATCH PAD FUNCS WITH LOW 4 BITS OF SP AS ADR
;2221            LOAD.LC=2                ;LOAD LC, ADR=SPO.RN
;2222            WRITE.RC=3               ;WRITE RC
;2223            LOAD.LAB=4               ;LOAD LA, LB
;2224            WRITE.RAB=5              ;WRITE RA, RB
;2225            LOAD.LAB1.WRITE.RC=6     ;LOAD LA, LB[R1], AND WRITE RC[RN]
;2226            LOAD.LC.WRITE.RAB1=7     ;LOAD LC[RN], AND WRITE RA, RB[R1]
```

```
;2227    .TOC    ''      Machine definition    : SP0.RAB, SP0.RC, SUB, VAK''
;2228
;2229    SP0.RAB/=<38:35>                ;RA/RB LOCATIONS
;2230            R0=0
;2231            R1=1
;2232            R2=2
;2233            R3=3
;2234            R4=4
;2235            R5=5
;2236            R6=6
;2237            R7=7
;2238            AP=0C                   ;R12 = ARGUMENT LIST POINTER
;2239            FP=0D                   ;R13 = STACK FRAME POINTER
;2240            SP=0E                   ;R14 = STACK POINTER
;2241            R15=0F                  ;R15 = PC, TO SOFTWARE, SCRATCH TO UCODE
;2242
;2243    SP0.RC/=<38:35>                 ;RC LOCATIONS
;2244            T0=0
;2245            T1=1
;2246            T2=2
;2247            T3=3
;2248            T4=4
;2249            T5=5
;2250            T6=6
;2251            T7=7
;2252            LC.SV=8                 ;MEM MGMT SAVES LC HERE
;2253            VA.SV=9
;2254            PTE.VA=0A
;2255            PTE.PA=0B
;2256            PC.SV=0C
;2257            SC.SV=0D
;2258            VA.REF=0E
;2259            MBIT.VA=0F
;2260            PTE.MASK=0F
;2261
;2262    SUB/=<65:64>,.DEFAULT=0         ;SUBROUTINE CONTROL
;2263            NOP=0                   ;DEFAULT
;2264            CALL=1                  ;PUSH UPC OF THIS MICROINSTRUCTION
;2265                                    ; ONTO USTACK
;2266            RET=2                   ;'OR'' TOP OF USTACK TO UPC
;2267                                    ; AND POP USTACK
;2268            SPEC=3                  ;REPLACE LOW 8 BITS OF NEXT
;2269                                    ; UPC WITH SPECIFIER DECODE FROM
;2270                                    ; INSTRUCTION BUFFER
;2271
;2272    VAK/=<25:25>,.DEFAULT=0
;2273            NOP=0                   ;DEFAULT
;2274            LOAD=1                  ;LOAD VA
```

```
;2275    .TOC    "       Machine definition    : Validity checks"
;2276
;2277            .SET/V0=<.NOT[<NATIVE>]>
;2278            .SET/V1=<NATIVE>
;2279
                                        ;2280   .BIN                     ;RE-ENABLE LISTING SPACE FOR BINARY OUTPUT
                                        ;2281   .CREF                    ;RE-ENABLE CROSS REFERENCE
                                        ;2282   .LIST                    ;Re-enable full listing
```

```
                                        ;2283   .TOC    'MACRO.MIC''
                                        ;2284   .TOC    'Revision 2.16''
                                        ;2285   ;        P. R. Guilbault
                                        ;2286
;2287   .NOBIN
;2288   .TOC    ''      Revision History''
;2289
;2290   ; 02    Add variable region expressions
;2291   ;       Add macro to support floating faults
;2292   ;       Add macro to support Patch no. 088
;2293   ;       Add macro to support Patch no. 093
;2294   ;       Increase WCS region to 7k to make room for G & H
;2295   ;       Change 'SET.N.AND.Z'' to 'N_AMX.Z_TST''
;2296   ;       Change 'SET.N&Z'' to 'N&Z_ALU.V&C_0''
;2297   ;       Add 'N&Z_ALU''
;2298   ;       Add 'PLS<C>_AMX0''
;2299   ;       Resort macros
;2300   ;       Re-do regions to get G & H into 1 chunk
;2301   ; 01    Add macros to support new patches
;2302   ; 00    Put macro here that was found in MULD
;2303   ;       Put macro here that was found in CVTFI2
;2304   ;       Edit and add tabs to allign macro definitions.
;2305   ;       Add macros that were previously in MUL & MULP
;2306   ;       Add macro for ACBD fix
;2307   ;       Start of history
```

```
;2308    .TOC     ''       Macro definition      : Regions''
;2309
;2310                      ;  +-------+----+----------------------+
;2311                      ;  |     0 | 0k |                      |
;2312                      ;  |       |    |                      |
;2313                      ;  |       |    |                      |
;2314                      ;  |       |    |                      |
;2315                      ;  |       | to | PCS                  |
;2316                      ;  |       |    |                      |
;2317                      ;  |       |    |                      |
;2318                      ;  |       |    |                      |
;2319                      ;  |  4095 | 4k |                      |
;2320                      ;  +-------+----+----------------------+
;2321    .SET/WCSR1L=1000  ;  |  4096 | 4k |                      |
;2322    .SET/WCSR1H=113F  ;  |       |    |                      |
;2323                      ;  |       | to | DEC'S WCS region     |
;2324    .SET/WCSR2L=1200  ;  |       |    |                      |
;2325    .SET/WCSR2H=17FF  ;  |  6143 | 6k |                      |
;2326                      ;  +-------+----+----------------------+
;2327    .SET/WCSR3L=1800  ;  |  6144 | 6k |                      |
;2328                      ;  |       |    |                      |
;2329                      ;  |       | to | User or G&H WCS      |
;2330                      ;  |       |    |                      |
;2331    .SET/WCSR3H=1BFF  ;  |  7167 | 7k |                      |
;2332                      ;  +-------+----+----------------------+
;2333    ;          1C00   ;  |  7168 | 7k |                      |
;2334                      ;  |       |    |                      |
;2335                      ;  |       | to | User WCS             |
;2336                      ;  |       |    |                      |
;2337                      ;  |  8191 | 8k |                      |
;2338    ;          1FFF   ;  +-------+----+----------------------+
```

Note : 1140 to 11FF is the FPLA trap address region

```
;2339   .TOC    ''              definition      : Register transfer macros''
;2340
;2341   ALU_-1                          ''AMX/RAMX.0XT,DT/LONG,ALU/NOTA''
;2342   ALU_0(A)                        ''AMX/RAMX.0XT,DT/LONG,ALU/A''
;2343   ALU_0+D                         ''AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A+B''
;2344   ALU_0+D+1                       ''AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A+B+1''
;2345   ALU_0+K[]                       ''KMX/a1,BMX/KMX,AMX/RAMX.0XT,DT/LONG,ALU/A+B''
;2346   ALU_0+K[]+1                     ''KMX/a1,BMX/KMX,AMX/RAMX.0XT,DT/LONG,ALU/A+B+1''
;2347   ALU_0+LB+1                      ''AMX/RAMX.0XT,DT/LONG,BMX/LB,ALU/A+B+1''
;2348   ALU_0+LC                        ''AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B''
;2349   ALU_0+LC+1                      ''AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B+1''
;2350   ALU_0+MASK+1                    ''AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1''
;2351   ALU_0+Q                         ''AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B''
;2352   ALU_0+Q+1                       ''AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B+1''
;2353   ALU_0-D                         ''AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B''
;2354   ALU_0-D-1                       ''AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B-1''
;2355   ALU_0-K[]                       ''AMX/RAMX.0XT,DT/LONG,KMX/a1,BMX/KMX,ALU/A-B''
;2356   ALU_0-K[]-1                     ''KMX/a1,BMX/KMX,AMX/RAMX.0XT,DT/LONG,ALU/A-B-1''
;2357   ALU_0-LB                        ''AMX/RAMX.0XT,DT/LONG,BMX/LB,ALU/A-B''
;2358   ALU_0-LC                        ''AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A-B''
;2359   ALU_0-LC-1                      ''AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A-B-1''
;2360   ALU_0-Q                         ''AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B''
;2361   ALU_0-Q-1                       ''AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B-1''
;2362   ALU_0[]D                        ''ALU/a1,AMX/RAMX.0XT,LONG,BMX/RBMX,RBMX/D''
;2363   ALU_0[]LC                       ''ALU/a1,AMX/RAMX.0XT,LONG,BMX/LC''
;2364   ALU_D                           ''RAMX/D,AMX/RAMX,ALU/A''
;2365   ALU_D(B)                        ''RBMX/D,BMX/RBMX,ALU/B''
;2366   ALU_D+K[]                       ''RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B''
;2367   ALU_D+K[]+1                     ''RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B+1''
;2368   ALU_D+K[].RLOG                  ''AMX/RAMX,RAMX/D,KMX/a1,BMX/KMX,ALU/A+B.RLOG''
;2369   ALU_D+LB                        ''RAMX/D,AMX/RAMX,BMX/LB,ALU/A+B''
;2370   ALU_D+LC                        ''RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B''
;2371   ALU_D+LC+1                      ''RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+1''
;2372   ALU_D+LC+PSL.C                  ''RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+PSL.C''
;2373   ALU_D+Q                         ''RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B''
;2374   ALU_D+Q+1                       ''RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1''
;2375   ALU_D+Q+PSL.C                   ''ALU/A+B+PSL.C,AMX/RAMX,BMX/RBMX,RBMX/Q,RAMX/D''
;2376   ALU_D+RLOG                      ''ALU/A+B,AMX/RAMX,RAMX/D,BMX/0,MSC/READ,RLOG''
;2377   ALU_D-K[]                       ''RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A-B''
;2378   ALU_D-K[]-1                     ''RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A-B-1''
;2379   ALU_D-LB                        ''RAMX/D,AMX/RAMX,BMX/LB,ALU/A-B''
;2380   ALU_D-LB.RLOG                   ''RAMX/D,AMX/RAMX,BMX/LB,ALU/A-B.RLOG''
;2381   ALU_D-LC                        ''RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B''
;2382   ALU_D-LC-1                      ''RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B-1''
;2383   ALU_D-Q                         ''RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B''
;2384   ALU_D-Q-1                       ''RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B-1''
;2385   ALU_D.OXT[]                     ''RAMX/D,AMX/RAMX.0XT,DT/a1,ALU/A''
;2386   ALU_D.OXT[]+K[]                 ''RAMX/D,AMX/RAMX.0XT,DT/a1,KMX/a2,BMX/KMX,ALU/A+B''
;2387   ALU_D.OXT[]+LC                  ''ALU/A+B,AMX/RAMX.0XT,DT/a1,RAMX/D,BMX/LC''
;2388   ALU_D.OXT[]+Q                   ''ALU/A+B,AMX/RAMX.0XT,DT/a1,RAMX/D,BMX/RBMX,RBMX/Q''
;2389   ALU_D.OXT[]-K[]                 ''RAMX/D,AMX/RAMX.0XT,DT/a1,KMX/a2,BMX/KMX,ALU/A-B''
;2390   ALU_D.OXT[]-Q                   ''RAMX/D,AMX/RAMX.0XT,DT/a1,RBMX/Q,BMX/RBMX,ALU/A-B''
;2391   ALU_D.OXT[].AND.K[]             ''RAMX/D,AMX/RAMX.0XT,DT/a1,KMX/a2,BMX/KMX,ALU/AND''
;2392   ALU_D.OXT[].ANDNOT.K[]          ''ALU/ANDNOT,AMX/RAMX.0XT,DT/a1,RAMX/D,BMX/KMX,KMX/a2''
;2393   ALU_D.OXT[].OR.Q                ''RAMX/D,AMX/RAMX.0XT,DT/a1,BMX/RBMX,ALU/OR''
```

```
;2394    ALU_D.AND.K[]                   'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND''
;2395    ALU_D.AND.MASK                  'RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND''
;2396    ALU_D.ANDNOT.K[]                'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT''
;2397    ALU_D.ANDNOT.MASK               'RAMX/D,AMX/RAMX,BMX/MASK,ALU/ANDNOT''
;2398    ALU_D.ANDNOT.Q                  'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT''
;2399    ALU_D.OR.K[]                    'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR''
;2400    ALU_D.OR.LC                     'RAMX/D,AMX/RAMX,BMX/LC,ALU/OR''
;2401    ALU_D.OR.Q                      'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR''
;2402    ALU_D.OR.RC[]                   'RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR''
;2403    ALU_D.ORNOT.MASK                'RAMX/D,AMX/RAMX,BMX/MASK,ALU/ORNOT''
;2404    ALU_D.SXT[]                     'RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A''
;2405    ALU_D.SXT[]+K[]                 'RAMX/D,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B''
;2406    ALU_D.SXT[]+Q                   'RAMX/D,AMX/RAMX.SXT,DT/@1,BMX/RBMX,ALU/A+B''
;2407    ALU_D.SXT[].ANDNOT.K[]          'RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/ANDNOT,BMX/KMX,KMX/@2''
;2408    ALU_D.SXT[].AND.K[]             'RAMX/D,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/AND''
;2409    ALU_D.XOR.K[]                   'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR''
;2410    ALU_D.XOR.LC                    'RAMX/D,AMX/RAMX,BMX/LC,ALU/XOR''
;2411    ALU_D.XOR.Q                     'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/XOR''
;2412    ALU_D.XOR.RC[]                  'RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR''
;2413    ALU_D.XOR.R[]                   'RAMX/D,AMX/RAMX,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/LB,ALU/XOR''
;2414    ALU_D[]K[]                      'RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/@1''
;2415    ALU_D[]LB                       'ALU/@1,AMX/RAMX,RAMX/D,BMX/LB''
;2416    ALU_D[]LC                       'RAMX/D,AMX/RAMX,BMX/LC,ALU/@1''
;2417    ALU_D[]Q                        'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1''
;2418    ALU_K[]                         'KMX/@1,BMX/KMX,ALU/B''
;2419    ALU_LA                          'AMX/LA,ALU/A''
;2420    ALU_LA+K[]                      'AMX/LA,KMX/@1,BMX/KMX,ALU/A+B''
;2421    ALU_LA+K[]+1                    'ALU/A+B+1,AMX/LA,BMX/KMX,KMX/@1''
;2422    ALU_LA+K[].RLOG                 'AMX/LA,KMX/@1,BMX/KMX,ALU/A+B.RLOG''
;2423    ALU_LA+LB                       'AMX/LA,BMX/LB,ALU/A+B''
;2424    ALU_LA+LC                       'ALU/A+B,AMX/LA,BMX/LC''
;2425    ALU_LA+LC+1                     'ALU/A+B+1,AMX/LA,BMX/LC''
;2426    ALU_LA+LC+PSL.C                 'ALU/A+B+PSL.C,AMX/LA,BMX/LC''
;2427    ALU_LA+Q                        'ALU/A+B,AMX/LA,BMX/RBMX,RBMX/Q''
;2428    ALU_LA-D                        'AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B''
;2429    ALU_LA-D-1                      'AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B-1''
;2430    ALU_LA-K[]                      'AMX/LA,KMX/@1,BMX/KMX,ALU/A-B''
;2431    ALU_LA-K[]-1                    'AMX/LA,KMX/@1,BMX/KMX,ALU/A-B-1''
;2432    ALU_LA-K[].RLOG                 'AMX/LA,KMX/@1,BMX/KMX,ALU/A-B.RLOG''
;2433    ALU_LA-LC                       'ALU/A-B,AMX/LA,BMX/LC''
;2434    ALU_LA-Q                        'ALU/A-B,AMX/LA,BMX/RBMX,RBMX/Q''
;2435    ALU_LA-Q-1                      'ALU/A-B-1,AMX/LA,BMX/RBMX,RBMX/Q''
;2436    ALU_LA AND.K[]                  'AMX/LA,KMX/@1,BMX/KMX,ALU/AND''
;2437    ALU_LA.AND.LC                   'ALU/AND,AMX/LA,BMX/LC''
;2438    ALU_LA.ANDNOT.K[]               'AMX/LA,KMX/@1,BMX/KMX,ALU/ANDNOT''
;2439    ALU_LA.ANDNOT.MASK              'AMX/LA,BMX/MASK,ALU/ANDNOT''
;2440    ALU_LA.OR.K[]                   'ALU/OR,AMX/LA,BMX/KMX,KMX/@1''
;2441    ALU_LA.XOR.LC                   'AMX/LA,BMX/LC,ALU/XOR''
;2442    ALU_LA[]D                       'AMX/LA,RBMX/D,BMX/RBMX,ALU/@1''
;2443    ALU_LA[]LB                      'AMX/LA,BMX/LB,ALU/@1''
;2444    ALU_LA[]Q                       'AMX/LA,RBMX/Q,BMX/RBMX,ALU/@1''
;2445    ALU_LB                          'BMX/LB,ALU/B''
;2446    ALU_LC                          'BMX/LC,ALU/B''
;2447    ALU_NOT.D                       'ALU/NOT,AMX/RAMX,RAMX/D''
;2448    ALU_NOT.K[]                     'BMX/KMX,KMX/@1,ALU/ORNOT,AMX/RAMX.OXT,DT/LONG''
```

```
;2449   ALU_NOT.RC[]                 "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT"
;2450   ALU_PACK.FP                  "BMX/PACKED.FL,ALU/B"
;2451   ALU_PC                       "BMX/PC,ALU/B"
;2452   ALU_Q                        "RAMX/Q,AMX/RAMX,ALU/A"
;2453   ALU_Q(B)                     "RBMX/Q,BMX/RBMX,ALU/B"
;2454   ALU_Q+K[]                    "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B"
;2455   ALU_Q+K[]+1                  "ALU/A+B+1,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@1"
;2456   ALU_Q+LB                     "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B"
;2457   ALU_Q+LB+1                   "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B+1"
;2458   ALU_Q+LC                     "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B"
;2459   ALU_Q+LC+1                   "ALU/A+B+1,AMX/RAMX,RAMX/Q,BMX/LC"
;2460   ALU_Q+LC+PSL.C               "ALU/A+B+PSL.C,AMX/RAMX,RAMX/Q,BMX/LC"
;2461   ALU_Q+MASK                   "ALU/A+B,AMX/RAMX,RAMX/Q,BMX/MASK"
;2462   ALU_Q-D                      "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B"
;2463   ALU_Q-D-1                    "ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D"
;2464   ALU_Q-K[]                    "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B"
;2465   ALU_Q-LB                     "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A-B"
;2466   ALU_Q-LC                     "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B"
;2467   ALU_Q-MASK-1                 "ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/MASK"
;2468   ALU_Q.OXT[]                  "RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A"
;2469   ALU_Q.OXT[]+D                "ALU/A+B,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RBMX/D,RAMX/Q"
;2470   ALU_Q.OXT[]+D+1              "ALU/A+B+1,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RAMX/Q,RBMX/D"
;2471   ALU_Q.OXT[]+K[]              "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
;2472   ALU_Q.OXT[]-D                "ALU/A-B,RAMX/Q,AMX/RAMX.OXT,DT/@1,BMX/RBMX"
;2473   ALU_Q.OXT[]-K[]              "ALU/A-B,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
;2474   ALU_Q.OXT[].ANDNOT.K[]       "ALU/ANDNOT,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
;2475   ALU_Q.OXT[].OR.K[]           "ALU/OR,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
;2476   ALU_Q.OXT[].OR.D             "ALU/OR,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/RBMX,RBMX/D"
;2477   ALU_Q.AND.D                  "AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D,ALU/AND"
;2478   ALU_Q.AND.K[]                "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND"
;2479   ALU_Q.ANDNOT.K[]             "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT"
;2480   ALU_Q.ANDNOT.MASK            "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ANDNOT"
;2481   ALU_Q.ANDNOT.R[]             "ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/LB,SPO.R/LOAD.LAB,SPO.RAB/@1"
;2482   ALU_Q.OR.K[]                 "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR"
;2483   ALU_Q.OR.LC                  "RAMX/Q,AMX/RAMX,BMX/LC,ALU/OR"
;2484   ALU_Q.ORNOT.K[]              "ALU/ORNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@1"
;2485   ALU_Q.SXT[]                  "ALU/A,AMX/RAMX.SXT,DT/@1,RAMX/Q"
;2486   ALU_Q.SXT[]+K[]              "RAMX/Q,AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B"
;2487   ALU_Q.SXT[]+LB               "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/LB,ALU/A+B"
;2488   ALU_Q.SXT[]+LB+1             "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/LB,ALU/A+B+1"
;2489   ALU_Q.SXT[]+PC               "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/PC,ALU/A+B"
;2490   ALU_Q.SXT[].ANDNOT.K[]       "ALU/ANDNOT,AMX/RAMX.SXT,RAMX/Q,BMX/KMX,KMX/@2,DT/@1"
;2491   ALU_Q.XOR.D                  "RAMX/Q,AMX/RAMX,BMX/RBMX,RBMX/D,ALU/XOR"
;2492   ALU_Q.XOR.K[]                "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR"
;2493   ALU_Q.XOR.LC                 "RAMX/Q,AMX/RAMX,BMX/LC,ALU/XOR"
;2494   ALU_Q.XOR.RC[]               "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR"
;2495   ALU_Q[]D                     "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/@1"
;2496   ALU_R(DST)                   "SPO.AC/LOAD.LAB,SPO.ACN11/DST.DST,AMX/LA,ALU/A"
;2497   ALU_R(SC).ANDNOT.K[]         "SPO.AC/LOAD.LAB,SPO.ACN/SC,AMX/LA,KMX/@1,BMX/KMX,ALU/ANDNOT"
;2498   ALU_R(SP1)+K[].RLOG          "SPO.AC/LOAD.LAB,SPO.ACN/SF1.SP1,AMX/LA,KMX/@1,BMX/KMX,ALU/A+B.RLOG"
;2499   ALU_RC(SC)                   "SPO/LOAD.LC.SC,BMX/LC,ALU/B"
;2500   ALU_RC[]                     "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B"
;2501   ALU_RLOG                     "BMX/0,ALU/B,MSC/READ.RLOG"
;2502   ALU_R[]                      "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A"
;2503   ALU_R[]-K[]                  "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/A-B"
```

```
;2504   ALU_R[].AND.K[]                 'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND''
;2505   ALU_R[].AND.LC                  'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/LC,ALU/AND''
;2506   ALU_R[].ANDNOT.K[]              'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/ANDNOT''
;2507   ALU_R[].ANDNOT.MASK             'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,BMX/MASK,ALU/ANDNOT''
;2508   ALU_R[].OR.K[]                  'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/OR''
;2509   ALU_R[].ORNOT.K[]               'ALU/ORNOT,AMX/LA,BMX/KMX,SPO.R/LOAD.LAB,SPO.RAB/@1,KMX/@2''
;2510   ALU_R[].XOR.K[]                 'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/XOR''
;2511
;2512   CACHE.P_D[]                     'VAK/NOP,MCT/WRITE.P,DT/@1,DK/NOP''
;2513   CACHE[]_D                       'VAK/NOP,MCT/WRITE.V.WCHK,MSC/@1,DK/NOP''
;2514   CACHE_D(QUAD)                   'MCT/EXTWRITE.P,LONG,VAK/NOP,DK/NOP''
;2515   CACHE_D.INST.DEP                'VAK/NOP,MCT/WRITE.V.WCHK,DT/INST.DEP,DK/NOP''
;2516   CACHE_D[]                       'VAK/NOP,MCT/WRITE.V.WCHK,DT/@1,DK/NOP''
;2517   CACHE_D[].LK                    'VAK/NOP,MCT/LOCKWRITE.V.XCHK,DT/@1,DK/NOP''
;2518   CACHE_D[].NOCHK                 'VAK/NOP,MCT/WRITE.V.NOCHK,DT/@1,DK/NOP''
;2519
;2520   D&Q_D+Q                         'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF,QK/SHF''
;2521   D&RC[]_PC                       'BMX/PC,ALU/B,SHF/ALU,DK/SHF,SPO.R/WRITE.RC,SPO.RC/@1''
;2522   D&VA_ALU                        'VAK/LOAD,SHF/ALU,DK/SHF''
;2523   D&VA_D+LC                       'RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,DK/SHF''
;2524   D&VA_D+Q                        'D_D+Q,VAK/LOAD''
;2525   D&VA_D-K[]                      'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,SHF/ALU,DK/SHF''
;2526   D&VA_LA                         'AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,DK/SHF''
;2527   D&VA_LB                         'BMX/LB,ALU/B,VAK/LOAD,SHF/ALU,DK/SHF''
;2528   D&VA_Q                          'RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,DK/Q''
;2529   D&VA_Q+LB.PC                    'RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,DK/SHF''
;2530
;2531   D[]_CACHE                       'VAK/NOP,MCT/READ.V.RCHK,DT/@1,DK/NOP''
;2532   D[]_CACHE.IBCHK                 'VAK/NOP,MCT/READ.V.IBCHK,DT/@1,DK/NOP''
;2533   D[]_CACHE.LK                    'VAK/NOP,MCT/LOCKREAD.V.WCHK,DT/@1,DK/NOP''
;2534   D[]_CACHE.NOCHK                 'VAK/NOP,MCT/READ.V.NOCHK,DT/@1,DK/NOP''
;2535   D[]_CACHE.P                     'VAK/NOP,MCT/READ.P,DT/@1,DK/NOP''
;2536   D[]_CACHE.WCHK                  'VAK/NOP,MCT/READ.V.WCHK,DT/@1,DK/NOP''
;2537
;2538   D_0                             'DK/CLR''
;2539   D_0+K[]+1                       'AMX/RAMX.0XT,DT/LONG,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF''
;2540   D_0+LC+1                        'AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,DK/SHF''
;2541   D_0-D                           'AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF''
;2542   D_0-K[]                         'AMX/RAMX.0XT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF''
;2543   D_0-Q                           'AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF''
;2544   D_0-Q-1                         'ALU_0-Q-1,D_ALU''
;2545   D_ACCEL&SYNC                    'DK/ACCEL,ACF/SYNC''
;2546   D_ALU                           'SHF/ALU,DK/SHF''
;2547   D_ALU(FRAC)                     'SHF/ALU,DK/SHF.FL''
;2548   D_ALU.LEFT                      'SHF/LEFT,DK/SHF''
;2549   D_ALU.LEFT2                     'SHF/ALU.DT,DT/LONG,DK/SHF''
;2550   D_ALU.LEFT3                     'SHF/LEFT3,DK/SHF''
;2551   D_ALU.RIGHT                     'SHF/RIGHT,DK/SHF''
;2552   D_ALU.RIGHT2                    'SHF/RIGHT2,DK/SHF''
;2553   D_BLANK                         'D_K[.20]''
;2554   D_CACHE.INST.DEP                'VAK/NOP,MCT/READ.V.IBCHK,DT/INST.DEP,DK/NOP''
;2555   D_CACHE.LK[]                    'VAK/NOP,MCT/LOCKREAD.V.WCHK,MSC/@1,DK/NOP''
;2556   D_CACHE.WCHK[]                  'VAK/NOP,MCT/READ.V.WCHK,MSC/@1,DK/NOP''
;2557   D_CACHE[]                       'VAK/NOP,MCT/READ.V.RCHK,MSC/@1,DK/NOP''
;2558   D_D(FRAC)                       'RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,DK/SHF.FL''
```

```
;2559   D_D+K[]                    'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF''
;2560   D_D+K[]+1                  'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF''
;2561   D_D+LB                     'RAMX/D,AMX/RAMX,BMX/LB,ALU/A+B,SHF/ALU,DK/SHF''
;2562   D_D+LC                     'RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,DK/SHF''
;2563   D_D+LC+PSL C               'RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+PSL.C,SHF/ALU,DK/SHF''
;2564   D_D+Q                      'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF''
;2565   D_D+Q+1                    'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1,SHF/ALU,DK/SHF''
;2566   D_D-K[]                    'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF''
;2567   D_D-LC                     'RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,DK/SHF''
;2568   D_D-Q                      'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF''
;2569   D_D-Q-1                    'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B-1,SHF/ALU,DK/SHF''
;2570   D_D.OXT[]                  'RAMX/D,AMX/RAMX.OXT,DT/a1,ALU/A,SHF/ALU,DK/SHF''
;2571   D_D.OXT[]+K[]              'RAMX/D,AMX/RAMX.OXT,DT/a1,KMX/a2,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF''
;2572   D_D.OXT[]+Q                'ALU/A+B,AMX/RAMX.OXT,DT/a1,BMX/RBMX,RBMX/Q,D_ALU''
;2573   D_D.OXT[]+Q+1              'RAMX/D,AMX/RAMX.OXT,DT/a1,BMX/RBMX,ALU/A+B+1,D_ALU''
;2574   D_D.OXT[].ANDNOT.K[]       'RAMX/D,AMX/RAMX.OXT,DT/a1,KMX/a2,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF''
;2575   D_D.OXT[].OR.Q             'RAMX/D,AMX/RAMX.OXT,DT/a1,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,DK/SHF''
;2576   D_D.OXT[].XOR.Q            'DK/SHF,ALU/XOR,SHF/ALU,AMX/RAMX.OXT,RAMX/D,DT/a1,RBMX/Q,BMX/RBMX''
;2577   D_D.OXT[].XOR.RC[]         'RAMX/D,AMX/RAMX.OXT,DT/a1,SPO.R/LOAD.LC,SPO.RC/a2,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF''
;2578   D_D.AND.K[]                'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF''
;2579   D_D.AND.K[].LEFT2          'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/AND,SHF/ALU.DT,DT/LONG,DK/SHF''
;2580   D_D.AND.K[].RIGHT          'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/AND,SHF/RIGHT,DK/SHF''
;2581   D_D.AND.LC                 'RAMX/D,AMX/RAMX,BMX/LC,ALU/AND,SHF/ALU,DK/SHF''
;2582   D_D.AND.MASK               'RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,DK/SHF''
;2583   D_D.AND.Q                  'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/AND,SHF/ALU,DK/SHF''
;2584   D_D.AND.RC[]               'RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/a1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF''
;2585   D_D.ANDNOT.K[]             'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF''
;2586   D_D.ANDNOT.LC              'RAMX/D,AMX/RAMX,BMX/LC,ALU/ANDNOT,SHF/ALU,DK/SHF''
;2587   D_D.ANDNOT.PSWZ            'DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.4,SHF/ALU''
;2588   D_D.ANDNOT.Q               'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,DK/SHF''
;2589   D_D.ANDNOT.RC[]            'RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/a1,BMX/LC,ALU/ANDNOT,SHF/ALU,DK/SHF''
;2590   D_D.LEFT                   'DK/LEFT''
;2591   D_D.LEFT2                  'DK/LEFT2''
;2592   D_D.OR.ASCII               'D_D.OR.K[.30]''
;2593   D_D.OR.K[]                 'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF''
;2594   D_D.OR.PSWC                'DK/SHF,ALU/OR,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.1,SHF/ALU''
;2595   D_D.OR.PSWV                'DK/SHF,ALU/OR,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.2,SHF/ALU''
;2596   D_D.OR.Q                   'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,DK/SHF''
;2597   D_D.OR.RC[]                'RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/a1,BMX/LC,ALU/OR,SHF/ALU,DK/SHF''
;2598   D_D.OR.R[]                 'ALU/OR,AMX/RAMX,RAMX/D,BMX/LB,SPO.R/LOAD.LAB,SPO.RAB/a1,DK/SHF''
;2599   D_D.ORNOT.MASK             'RAMX/D,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,DK/SHF''
;2600   D_D.RIGHT                  'DK/RIGHT''
;2601   D_D.RIGHT(B)               'RBMX/D,BMX/RBMX,ALU/B,SHF/RIGHT,DK/SHF''
;2602   D_D.RIGHT2                 'DK/RIGHT2''
;2603   D_D.SWAP                   'DK/BYTE.SWAP''
;2604   D_D.SXT[]                  'RAMX/D,AMX/RAMX.SXT,DT/a1,ALU/A,SHF/ALU,DK/SHF''
;2605   D_D.SXT[].RIGHT            'RAMX/D,AMX/RAMX.SXT,DT/a1,ALU/A,SHF/RIGHT,DK/SHF''
;2606   D_D.XOR.K[]                'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/XOR,SHF/ALU,DK/SHF''
;2607   D_D.XOR.LC                 'RAMX/D,AMX/RAMX,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF''
;2608   D_D.XOR.Q                  'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/XOR,SHF/ALU,DK/SHF''
;2609   D_DAL.NORM                 'DK/DAL.SV''
;2610   D_DAL.SC                   'DK/DAL.SC''
;2611   D_D[]K[]                   'RAMX/D,AMX/RAMX,KMX/a2,BMX/KMX,ALU/a1,SHF/ALU,DK/SHF''
;2612   D_D[]MASK                  'RAMX/D,AMX/RAMX,BMX/MASK,ALU/a1,SHF/ALU,DK/SHF''
;2613   D_D[]Q                     'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/a1,SHF/ALU,DK/SHF''
```

```
;2614   D_INT.SUM                       'MCT/READ.INT.SUM,DK/NOP''
;2615   D_K[]                           'KMX/@1,BMX/KMX,ALU/B,SHF/ALU,DK/SHF''
;2616   D_K[].RIGHT                     'KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT,DK/SHF''
;2617   D_K[].RIGHT2                    'KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT2,DK/SHF''
;2618   D_LA                            'AMX/LA,ALU/A,SHF/ALU,DK/SHF''
;2619   D_LA(FRAC)                      'AMX/LA,ALU/A,SHF/ALU,DK/SHF.FL''
;2620   D_LA+D+PSL.C                    'AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B+PSL.C,SHF/ALU,DK/SHF''
;2621   D_LA-D                          'DK/SHF,ALU/A-B,AMX/LA,BMX/RBMX,RBMX/D,SHF/ALU''
;2622   D_LA-K[]                        'AMX/LA,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF''
;2623   D_LA.AND.K[]                    'AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF''
;2624   D_LA.RIGHT                      'AMX/LA,ALU/A,SHF/RIGHT,DK/SHF''
;2625   D_LB                            'BMX/LB,ALU/B,SHF/ALU,DK/SHF''
;2626   D_LB.PC                         'BMX/PC.OR.LB,ALU/B,SHF/ALU,DK/SHF''
;2627   D_LC                            'BMX/LC,ALU/B,SHF/ALU,DK/SHF''
;2628   D_LC(FRAC)                      'BMX/LC,ALU/B,SHF/ALU,DK/SHF.FL''
;2629   D_NOT.D                         'RAMX/D,AMX/RAMX,ALU/NOTA,SHF/ALU,DK/SHF''
;2630   D_NOT.K[]                       'KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT,SHF/ALU,DK/SHF''
;2631   D_NOT.MASK                      'BMX/MASK,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT,SHF/ALU,DK/SHF''
;2632   D_NOT.Q                         'RAMX/Q,AMX/RAMX,ALU/NOTA,SHF/ALU,DK/SHF''
;2633   D_NOT.R[]                       'LA_RA[@1],AMX/LA,ALU/NOTA,D_ALU''
;2634   D_PACK.FP                       'BMX/PACKED.FL,ALU/B,SHF/ALU,DK/SHF''
;2635   D_PACK.FP.LEFT                  'BMX/PACKED.FL,ALU/B,SHF/LEFT,DK/SHF''
;2636   D_PC                            'BMX/PC,ALU/B,SHF/ALU,DK/SHF''
;2637   D_PC.LEFT                       'BMX/PC,ALU/B,SHF/LEFT,DK/SHF''
;2638   D_Q                             'DK/Q''
;2639   D_Q(FRAC)                       'RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,DK/SHF.FL''
;2640   D_Q+D                           'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF''
;2641   D_Q+K[]                         'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF''
;2642   D_Q+LB                          'RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B,SHF/ALU,DK/SHF''
;2643   D_Q+PC                          'RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,DK/SHF''
;2644   D_Q-D                           'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF''
;2645   D_Q-D-1                         'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B-1,SHF/ALU,DK/SHF''
;2646   D_Q-K[]                         'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF''
;2647   D_Q-K[]-1                       'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1,SHF/ALU,DK/SHF''
;2648   D_Q-PCSV                        'RAMX/Q,AMX/RAMX,BMX/0,MSC/READ.PLOG,ALU/A-B,SHF/ALU,DK/SHF''
;2649   D_Q.OXT[]                       'RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/ALU,DK/SHF''
;2650   D_Q.AND.K[]                     'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF''
;2651   D_Q.AND.LC                      'RAMX/Q,AMX/RAMX,BMX/LC,ALU/AND,SHF/ALU,DK/SHF''
;2652   D_Q.AND.MASK                    'RAMX/Q,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,DK/SHF''
;2653   D_Q.AND.RC[]                    'RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF''
;2654   D_Q.ANDNOT.D                    'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/ANDNOT,SHF/ALU,DK/SHF''
;2655   D_Q.ANDNOT.K[]                  'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF''
;2656   D_Q.ANDNOT.MASK                 'RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ANDNOT,SHF/ALU,DK/SHF''
;2657   D_Q.ANDNOT.PSWC                 'DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.1,SHF/ALU''
;2658   D_Q.ANDNOT.PSWN                 'DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.8,SHF/ALU''
;2659   D_Q.ANDNOT.PSWZ                 'DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.4,SHF/ALU''
;2660   D_Q.LEFT                        'RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT,DK/SHF''
;2661   D_Q.OR.K[]                      'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF''
;2662   D_Q.OR.PSWC                     'DK/SHF,ALU/OR,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.1,SHF/ALU''
;2663   D_Q.OR.RC[]                     'RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,DK/SHF''
;2664   D_Q.ORNOT.MASK                  'RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,DK/SHF''
;2665   D_Q.RIGHT                       'RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT,DK/SHF''
;2666   D_Q.RIGHT2                      'RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT2,DK/SHF''
;2667   D_Q.SXT[]                       'RAMX/Q,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,DK/SHF''
;2668   D_Q.XOR.RC[]                    'RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF''
```

```
;2669    D_Q[]D                          'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/@1,SHF/ALU,DK/SHF''
;2670    D_Q[]K[]                        'ALU/@1,SHF/ALU,DK/SHF,BMX/KMX,KMX/@2,AMX/RAMX,RAMX/Q''
;2671    D_Q[]MASK                       'RAMX/Q,AMX/RAMX,BMX/MASK,ALU/@1,SHF/ALU,DK/SHF''
;2672    D_R(PRN+1)                      'SPO.AC/LOAD.LAB,SPO.ACN/PRN+1,AMX/LA,ALU/A,SHF/ALU,DK/SHF''
;2673    D_R(SC)                         'SPO.AC/LOAD.LAB,SPO.ACN/SC,AMX/LA,ALU/A,SHF/ALU,DK/SHF''
;2674    D_R(SP1+1)                      'SPO.AC/LOAD.LAB,SPO.ACN/SP1+1,AMX/LA,ALU/A,SHF/ALU,DK/SHF''
;2675    D_RC(SC)                        'SPO/LOAD.LC.SC,BMX/LC,ALU/B,SHF/ALU,DK/SHF''
;2676    D_RC[]                          'SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SHF/ALU,DK/SHF''
;2677    D_RLOG                          'BMX/0,MSC/READ.RLOG,ALU/B,SHF/ALU,DK/SHF''
;2678    D_RLOG.RIGHT                    'BMX/0,MSC/READ.RLOG,ALU/B,SHF/RIGHT,DK/SHF''
;2679    D_R[]                           'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF''
;2680    D_R[](FRAC)                     'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF.FL''
;2681    D_R[].AND.K[]                   'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF''
;2682    D_R[].OR.K[]                    'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF''
;2683    D_R[].ORNOT.K[]                 'LAB_R[@1],AMX/LA,BMX/KMX,KMX/@2,ALU/ORNOT,D_ALU''
;2684
;2685    EALU_D(EXP)                     'RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B''
;2686    EALU_FE                         'EBMX/FE,EALU/B''
;2687    EALU_K[]                        'KMX/@1,EBMX/KMX,EALU/B''
;2688    EALU_R[](EXP)                   'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,EBMX/AMX.EXP,EALU/B''
;2689    EALU_SC                         'EALU/A''
;2690    EALU_SC+FE                      'EBMX/FE,EALU/A+B''
;2691    EALU_SC+K[]                     'KMX/@1,EBMX/KMX,EALU/A+B''
;2692    EALU_SC-FE                      'EBMX/FE,EALU/A-B''
;2693    EALU_SC-K[]                     'KMX/@1,EBMX/KMX,EALU/A-B''
;2694    EALU_SC.ANDNOT.K[]              'KMX/@1,EBMX/KMX,EALU/ANDNOT''
;2695    EALU_STATE                      'EALU/A,MSC/LOAD.STATE''
;2696
;2697    FE&SC_K[]                       'KMX/@1,EBMX/KMX,EALU/B,FEK/LOAD,SMX/EALU,SCK/LOAD''
;2698    FE_0(A)                         'AMX/RAMX.0XT,DT/LONG,EBMX/AMX.EXP,EALU/B,FEK/LOAD''
;2699    FE_D(EXP)                       'RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,FEK/LOAD''
;2700    FE_EALU                         'FEK/LOAD''
;2701    FE_K[]                          'KMX/@1,EBMX/KMX,EALU/B,FEK/LOAD''
;2702    FE_LA(EXP)                      'AMX/LA,EBMX/AMX.EXP,EALU/B,FEK/LOAD''
;2703    FE_NABS(SC-FE)                  'EALU/NABS.A-B,EBMX/FE,FEK/LOAD''
;2704    FE_NABS(SC-LA(EXP))             'AMX/LA,EBMX/AMX.EXP,EALU/NABS.A-B,FEK/LOAD''
;2705    FE_Q(EXP)                       'RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,FEK/LOAD''
;2706    FE_R[](EXP)                     'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,EBMX/AMX.EXP,EALU/B,FEK/LOAD''
;2707    FE_SC                           'EALU/A,FEK/LOAD''
;2708    FE_SC+1                         'EALU/A+1,FEK/LOAD''
;2709    FE_SC+FE                        'EBMX/FE,EALU/A+B,FEK/LOAD''
;2710    FE_SC+K[]                       'KMX/@1,EBMX/KMX,EALU/A+B,FEK/LOAD''
;2711    FE_SC+LA(EXP)                   'AMX/LA,EBMX/AMX.EXP,EALU/A+B,FEK/LOAD''
;2712    FE_SC-FE                        'EBMX/FE,EALU/A-B,FEK/LOAD''
;2713    FE_SC-K[]                       'KMX/@1,EBMX/KMX,EALU/A-B,FEK/LOAD''
;2714    FE_SC-LA(EXP)                   'AMX/LA,EBMX/AMX.EXP,EALU/A-B,FEK/LOAD''
;2715    FE_SC-SHF.VAL                   'EBMX/SHF.VAL,EALU/A-B,FEK/LOAD''
;2716    FE_SC.ANDNOT.FE                 'EBMX/FE,EALU/ANDNOT,FEK/LOAD''
;2717    FE_SC.ANDNOT.K[]                'KMX/@1,EBMX/KMX,EALU/ANDNOT,FEK/LOAD''
;2718    FE_SC.OR.K[]                    'EALU/OR,EBMX/KMX,KMX/@1,FEK/LOAD''
;2719    FE_SHF.VAL                      'EBMX/SHF.VAL,EALU/B,FEK/LOAD''
;2720    FE_STATE                        'MSC/LOAD.STATE,EALU/A,FEK/LOAD''
;2721
;2722    ID(SC)_D                        ''CID/WRITE.SC''
;2723    ID[]_D                          ''CID/WRITE.KMX,ID.ADDR/@1''
```

```
;2724   ID_D&NO.SYNC                    "CID/WRITE.KMX,ADS/IBA,KMX/SP1.CON"
;2725   ID_D.SYNC                       "CID/WRITE.KMX,ADS/IBA,KMX/SP1.CON,ACF/SYNC"
;2726
;2727   K[]                             "KMX/a1"
;2728
;2729   LAB_R(DST)                      "SPO.AC/LOAD.LAB,SPO.ACN11/DST.DST"
;2730   LAB_R(PRN)                      "SPO.AC/LOAD.LAB,SPO.ACN/PRN"
;2731   LAB_R(PRN+1)                    "SPO.AC/LOAD.LAB,SPO.ACN/PRN+1"
;2732   LAB_R(SC)                       "SPO.AC/LOAD.LAB,SPO.ACN/SC"
;2733   LAB_R(SP1)                      "SPO.AC/LOAD.LAB,SPO.ACN/SP1.SP1"
;2734   LAB_R(SP1+1)                    "SPO.AC/LOAD.LAB,SPO.ACN/SP1+1"
;2735   LAB_R1&RC[]_0                   "ALU_0(A),LAB_R1&RC[a1]_ALU"
;2736   LAB_R1&RC[]_0+LC+1              "ALU_A+B+1,AMX/RAMX.OXT_DT/LONG,BMX/LC,SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/a1,SHF/ALU"
;2737   LAB_R1&RC[]_0-D                 "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/a1,ALU/A-B,AMX/RAMX.OXT,DT/LONG,BMX/RBMX,RBMX/D,SHF/ALU"
;2738   LAB_R1&RC[]_ALU                 "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/a1,SHF/ALU"
;2739   LAB_R1&RC[]_ALU.RIGHT2          "SPO.R/LOAD.LAB1.WRITE.RC,SPO.RC/a1,SHF/RIGHT2"
;2740   LAB_R1&RC[]_D+LC                "ALU_D+LC,LAB_R1&RC[a1]_ALU"
;2741   LAB_R1&RC[]_D.OXT[]+K[]         "ALU_D.OXT[a2]+K[a3],LAB_R1&RC[a1]_ALU"
;2742   LAB_R1&RC[]_Q-K[]               "ALU_Q-K[a2],LAB_R1&RC[a1]_ALU"
;2743   LAB_RC[]                        "SPO.R/LOAD.LAB,SPO.RAB/a1"
;2744
;2745   LA_R(DST)&LB_R(SRC)             "SPO.AC/LOAD.LAB,SPO.ACN11/DST.SRC"
;2746   LA_R(SP2)&LB_R(SP1)             "SPO.AC/LOAD.LAB,SPO.ACN/SP2.SP1"
;2747   LA_RA[]                         "SPO.AC/LOAD.LA,SPO.RAB/a1"
;2748   LC_RC(SC)                       "SPO/LOAD.LC.SC"
;2749   LC_RC[]                         "SPO.R/LOAD.LC,SPO.RC/a1"
;2750   LC_RC[]&R1_(LA+LB).LEFT         "AMX/LA,BMX/LB,ALU/A+B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/a1"
;2751   LC_RC[]&R1_(LA+LB+PSL.C).LEFT   "AMX/LA,BMX/LB,ALU/A+B+PSL.C,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/a1"
;2752   LC_RC[]&R1_(LA+LB.RLOG).LEFT    "AMX/LA,BMX/LB,ALU/A+B.RLOG,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/a1"
;2753   LC_RC[]&R1_(LA-LB).LEFT         "AMX/LA,BMX/LB,ALU/A-B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/a1"
;2754   LC_RC[]&R1_(LA-LB.RLOG).LEFT    "AMX/LA,BMX/LB,ALU/A-B.RLOG,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/a1"
;2755   LC_RC[]&R1_ALU                  "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/a1,SHF/ALU"
;2756   LC_RC[]&R1_D                    "ALU_D,LC_RC[a1]&R1_ALU"
;2757   LC_RC[]&R1_LA+K[]               "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/a1,SHF/ALU,ALU/A+B,AMX/LA,BMX/KMX,KMX/a2"
;2758   LC_RC[]&R1_LA-K[]               "ALU_LA-K[a2],LC_RC[a1]&R1_ALU"
;2759   LC_RC[]&R1_LB                   "ALU_LB,LC_RC[a1]&R1_ALU"
;2760   LC_RC[]&R1_Q                    "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/a1,SHF/ALU,ALU/A,AMX/RAMX,RAMX/Q"
;2761
;2762   N&Z_ALU                         "CCK/NZ_ALU.VC_VC"
;2763   N&Z_ALU.V&C_0                   "CCK/NZ_ALU.VC_0"
;2764   N_AMX.Z_TST                     "CCK/N_AMX.Z_TST.VC_VC"
;2765
;2766   PC&VA_ALU                       "VAK/LOAD,PCK/PC_VA"
;2767   PC&VA_D                         "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
;2768   PC&VA_D+K[]                     "RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B,VAK/LOAD,PCK/PC_VA"
;2769   PC&VA_D-K[]                     "RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
;2770   PC&VA_D-PC                      "RAMX/D,AMX/RAMX,BMX/PC,ALU/A-B,VAK/LOAD,PCK/PC_VA"
;2771   PC&VA_D.OXT[]                   "RAMX/D,AMX/RAMX.OXT,DT/a1,ALU/A,VAK/LOAD,PCK/PC_VA"
;2772   PC&VA_D.OXT[]+PC                "RAMX/D,AMX/RAMX.OXT,DT/a1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
;2773   PC&VA_D.SXT[]+PC                "RAMX/D,AMX/RAMX.SXT,DT/a1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
;2774   PC&VA_K[]                       "KMX/a1,BMX/KMX,ALU/B,VAK/LOAD,PCK/PC_VA"
;2775   PC&VA_PC                        "BMX/PC,ALU/B,VAK/LOAD,PCK/PC_VA"
;2776   PC&VA_Q                         "RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
;2777   PC&VA_Q+PC                      "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
;2778   PC&VA_Q-D                       "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
```

```
;2779   PC&VA_Q-K[]                    'RAMX/Q,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A-B,VAK/LOAD,PCK/PC_VA''
;2780   PC&VA_Q.SXT[]+PC               'RAMX/Q,AMX/RAMX.SXT,DT/a1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA''
;2781   PC&VA_R[]                      'SPO.R/LOAD.LC,SPO.RC/a1,BMX/LC,ALU/B,VAK/LOAD,PCK/PC_VA''
;2782   PC&VA_R[].ANDNOT.K[]           'SPO.R/LOAD.LAB,SPO.RAB/a1,AMX/LA,KMX/a2,BMX/KMX,ALU/ANDNOT,VAK/LOAD,PCK/PC_VA''
;2783
;2784   PC_PC+1                        'PCK/PC+1''
;2785   PC_PC+2                        'PCK/PC+2''
;2786   PC_PC+4                        'PCK/PC+4''
;2787   PC_PC+N                        'PCK/PC+N''
;2788   PC_Q+PC                        'ALU/A+B,VAK/LOAD,PCK/PC_VA,BMX/PC,AMX/RAMX,RAMX/Q''
;2789   PC_VA                          'PCK/PC_VA''
;2790   PC_VIBA                        'PCK/PC_IBA''
;2791   PS[<C>_AMX0                    'CCK/C_AMX0''
;2792
;2793   Q&VA_ALU                       'VAK/LOAD,SHF/ALU,QK/SHF''
;2794   Q&VA_D                         'RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF''
;2795   Q&VA_D+LC                      'RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF''
;2796   Q&VA_LA                        'AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF''
;2797   Q&VA_Q+LB.PC                   'RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF''
;2798
;2799   QD_(Q+LB)D.RIGHT2              'ALU_Q+LB,Q_ALU.RIGHT2,D_D.RIGHT2''
;2800   QD_(Q+LC)D.RIGHT2              'ALU_Q+LC,Q_ALU.RIGHT2,D_D.RIGHT2''
;2801   QD_(Q-LB)D.RIGHT2              'ALU_Q-LB,Q_ALU.RIGHT2,D_D.RIGHT2''
;2802   QD_(Q-LC)D.RIGHT2              'ALU_Q-LC,Q_ALU.RIGHT2,D_D.RIGHT2''
;2803   QD_QD.RIGHT2                   'ALU_Q,Q_ALU.RIGHT2,D_D.RIGHT2''
;2804
;2805   Q_0                            'QK/CLR''
;2806   Q_0+LC+1                       'ALU/A+B+1,AMX/RAMX.0XT,DT/LONG,SHF/ALU,QK/SHF,BMX/LC''
;2807   Q_0+MASK+1                     'AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,QK/SHF''
;2808   Q_0+PC.RLOG                    'AMX/RAMX.0XT,DT/LONG,BMX/PC,ALU/A+B.RLOG,SHF/ALU,QK/SHF''
;2809   Q_0-D                          'AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF''
;2810   Q_0-K[]                        'AMX/RAMX.0XT,DT/LONG,KMX/a1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF''
;2811   Q_0-LC                         'AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF''
;2812   Q_0-Q                          'AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF''
;2813   Q_ACCEL&SYNC                   'QK/ACCEL,ACF/SYNC''
;2814   Q_ALU                          'SHF/ALU,QK/SHF''
;2815   Q_ALU(FRAC)                    'SHF/ALU,QK/SHF.FL''
;2816   Q_ALU.LEFT                     'SHF/LEFT,QK/SHF''
;2817   Q_ALU.LEFT2                    'SHF/ALU.DT,DT/LONG,QK/SHF''
;2818   Q_ALU.LEFT3                    'QK/SHF,SHF/LEFT3''
;2819   Q_ALU.RIGHT                    'SHF/RIGHT,QK/SHF''
;2820   Q_ALU.RIGHT2                   'SHF/RIGHT2,QK/SHF''
;2821   Q_D                            'QK/D''
;2822   Q_D(FRAC)(B)                   'RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,QK/SHF.FL''
;2823   Q_D+K[]                        'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF''
;2824   Q_D+K[]+1                      'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B+1,SHF/ALU,QK/SHF''
;2825   Q_D+K[].LEFT                   'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B,SHF/LEFT,QK/SHF''
;2826   Q_D+LC                         'RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF''
;2827   Q_D-K[]                        'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF''
;2828   Q_D-LC                         'RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF''
;2829   Q_D-Q                          'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF''
;2830   Q_D.0XT[]                      'RAMX/D,AMX/RAMX.0XT,DT/a1,ALU/A,SHF/ALU,QK/SHF''
;2831   Q_D.0XT[]+K[].LEFT             'RAMX/D,AMX/RAMX.0XT,DT/a1,KMX/a2,BMX/KMX,ALU/A+B,SHF/LEFT,QK/SHF''
;2832   Q_D.0XT[].OR.PACK.FP           'RAMX/D,AMX/RAMX.0XT,DT/a1,BMX/PACKED.FL,ALU/OR,QK/SHF''
;2833   Q_D.AND.K[]                    'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF''
```

```
;2834   Q_D.AND.K[].RIGHT              'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,QK/SHF''
;2835   Q_D.AND.K[].RIGHT2             'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT2,QK/SHF''
;2836   Q_D.AND.RC[]                   'RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,QK/SHF''
;2837   Q_D.ANDNOT.RC[]                'RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF''
;2838   Q_D.LEFT3                      'RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT3,QK/SHF''
;2839   Q_D.OR.K[]                     'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,QK/SHF''
;2840   Q_D.OR.RC[]                    'RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,QK/SHF''
;2841   Q_D.RIGHT                      'RAMX/D,AMX/RAMX,ALU/A,SHF/RIGHT,QK/SHF''
;2842   Q_D.RIGHT2                     'RAMX/D,AMX/RAMX,ALU/A,SHF/RIGHT2,QK/SHF''
;2843   Q_D.SXT[]                      'RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,QK/SHF''
;2844   Q_D.XOR.Q                      'QK/SHF,ALU/XOR,AMX/RAMX,RAMX/D,BMX/RBMX,RBMX/Q,SHF/ALU''
;2845   Q_DEC.CON                      'QK/DEC.CON''
;2846   Q_IB.BDEST                     'IBC/BDEST,QK/ID,MCT/ALLOW.IB.READ''
;2847   Q_IB.DATA                      'QK/ID,MCT/ALLOW.IB.READ''
;2848   Q_ID(SC)                       'CID/READ.SC,QK/ID''
;2849   Q_ID[]                         'CID/READ.KMX,ID.ADDR/@1,QK/ID''
;2850   Q_K[]                          'KMX/@1,BMX/KMX,ALU/B,SHF/ALU,QK/SHF''
;2851   Q_K[]+1                        'AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,QK/SHF''
;2852   Q_K[].CTX                      'KMX/@1,BMX/KMX,ALU/B,SHF/ALU.DT,DT/INST.DEP,QK/SHF''
;2853   Q_K[].RIGHT                    'KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT,QK/SHF''
;2854   Q_K[].RIGHT2                   'KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT2,QK/SHF''
;2855   Q_LA                           'AMX/LA,ALU/A,SHF/ALU,QK/SHF''
;2856   Q_LA+K[]                       'AMX/LA,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF''
;2857   Q_LA+Q                         'AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,QK/SHF''
;2858   Q_LA-K[]                       'AMX/LA,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF''
;2859   Q_LA.AND.K[]                   'AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF''
;2860   Q_LA.ANDNOT.RC[]               'AMX/LA,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF''
;2861   Q_LB                           'BMX/LB,ALU/B,SHF/ALU,QK/SHF''
;2862   Q_LC                           'BMX/LC,ALU/B,SHF/ALU,QK/SHF''
;2863   Q_NOT.Q                        'RAMX/Q,AMX/RAMX,ALU/NOTA,SHF/ALU,QK/SHF''
;2864   Q_NOT.R[]                      'LA_RA[@1],AMX/LA,ALU/NOTA,Q_ALU''
;2865   Q_PACK.FP                      'BMX/PACKED.FL,ALU/B,SHF/ALU,QK/SHF''
;2866   Q_PC                           'BMX/PC,ALU/B,SHF/ALU,QK/SHF''
;2867   Q_Q(FRAC)                      'RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,QK/SHF.FL''
;2868   Q_Q(FRAC)(B)                   'RBMX/Q,BMX/RBMX,ALU/B,SHF/ALU,QK/SHF.FL''
;2869   Q_Q+D                          'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,QK/SHF''
;2870   Q_Q+K[]                        'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF''
;2871   Q_Q+K[]+1                      'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,QK/SHF''
;2872   Q_Q+LC                         'RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF''
;2873   Q_Q+PC                         'RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,QK/SHF''
;2874   Q_Q-D                          'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF''
;2875   Q_Q-D-1                        'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B-1,SHF/ALU,QK/SHF''
;2876   Q_Q-K[]                        'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF''
;2877   Q_Q-K[]-1                      'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1,SHF/ALU,QK/SHF''
;2878   Q_Q-LC                         'RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF''
;2879   Q_Q-LC-1                       'RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B-1,SHF/ALU,QK/SHF''
;2880   Q_Q-MASK-1                     'RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU,QK/SHF''
;2881   Q_Q.OXT[]-K[]                  'RAMX/Q,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF''
;2882   Q_Q.OXT[].LEFT                 'RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/LEFT,QK/SHF''
;2883   Q_Q.OXT[].OR.D                 'RAMX/Q,AMX/RAMX.OXT,DT/@1,RBMX/D,BMX/RBMX,ALU/OR,SHF/ALU,QK/SHF''
;2884   Q_Q.AND.K[]                    'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF''
;2885   Q_Q.AND.K[].RIGHT2             'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT2,QK/SHF''
;2886   Q_Q.AND.K[].RIGHT              'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,QK/SHF''
;2887   Q_Q.AND.R[]                    'RAMX/Q,AMX/RAMX,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/LB,ALU/AND,SHF/ALU,QK/SHF''
;2888   Q_Q.AND.RC[]                   'RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,QK/SHF''
```

```
;2889    Q_Q.ANDNOT.D                    'RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/ANDNOT,SHF/ALU,QK/SHF''
;2890    Q_Q.ANDNOT.K[]                  'RAMX/Q,AMX/RAMX,KMX/a1,BMX/KMX,ALU/ANDNOT,SHF/ALU,QK/SHF''
;2891    Q_Q.ANDNOT.RC[]                 'RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/a1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF''
;2892    Q_Q.LEFT                        'QK/LEFT''
;2893    Q_Q.LEFT2                       'QK/LEFT2''
;2894    Q_Q.OR.K[]                      'RAMX/Q,AMX/RAMX,KMX/a1,BMX/KMX,ALU/OR,SHF/ALU,QK/SHF''
;2895    Q_Q.ORNOT.MASK                  'RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,QK/SHF''
;2896    Q_Q.RIGHT                       'QK/RIGHT''
;2897    Q_Q.RIGHT2                      'QK/RIGHT2''
;2898    Q_Q.SXT[]                       'RAMX/Q,AMX/RAMX.SXT,DT/a1,ALU/A,SHF/ALU,QK/SHF''
;2899    Q_Q.XOR.K[]                     'RAMX/Q,AMX/RAMX,KMX/a1,BMX/KMX,ALU/XOR,SHF/ALU,QK/SHF''
;2900    Q_R(PRN).ANDNOT.Q               'SPO.AC/LOAD.LAB,SPO.ACN/PRN,AMX/LA,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,QK/SHF''
;2901    Q_R(PRN+1)                      'SPO.AC/LOAD.LAB,SPO.ACN/PRN+1,AMX/LA,ALU/A,SHF/ALU,QK/SHF''
;2902    Q_R(PRN+1).AND.Q                'SPO.AC/LOAD.LAB,SPO.ACN/PRN+1,AMX/LA,RBMX/Q,BMX/RBMX,ALU/AND,SHF/ALU,QK/SHF''
;2903    Q_R(SC)                         'ALU/A,SHF/ALU,AMX/LA,SPO.AC/LOAD.LAB,SPO.ACN/SC,QK/SHF''
;2904    Q_R(SRC!1).AND.K[]              'SPO.AC/LOAD.LAB,SPO.ACN11/SRC.OR.1,AMX/LA,KMX/a1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF''
;2905    Q_RC(SC)                        'ALU/B,SHF/ALU,BMX/LC,SPO/LOAD.LC.SC,QK/SHF''
;2906    Q_RC[]                          'SPO.R/LOAD.LC,SPO.RC/a1,BMX/LC,ALU/B,SHF/ALU,QK/SHF''
;2907    Q_RC[](FRAC)                    'SPO.R/LOAD.LC,SPO.RC/a1,BMX/LC,ALU/B,SHF/ALU,QK/SHF.FL''
;2908    Q_R[]                           'SPO.R/LOAD.LAB,SPO.RAB/a1,AMX/LA,ALU/A,SHF/ALU,QK/SHF''
;2909    Q_R[](FRAC)                     'SPO.R/LOAD.LAB,SPO.RAB/a1,AMX/LA,ALU/A,SHF/ALU,QK/SHF.FL''
;2910    Q_R[].AND.K[]                   'SPO.R/LOAD.LAB,SPO.RAB/a1,AMX/LA,KMX/a2,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF''
;2911    Q_R[].AND.K[].RIGHT             'SPO.R/LOAD.LAB,SPO.RAB/a1,AMX/LA,ALU/AND,BMX/KMX,KMX/a2,SHF/RIGHT,QK/SHF''
;2912    Q_R[].ANDNOT.K[]                'SPO.R/LOAD.LAB,SPO.RAB/a1,AMX/LA,KMX/a2,BMX/KMX,ALU/ANDNOT,SHF/ALU,QK/SHF''
;2913    Q_R[].OR.K[]                    'ALU/OR,AMX/LA,SPO.R/LOAD.LAB,SPO.RAB/a1,BMX/KMX,KMX/a2,QK/SHF''
;2914    Q_SC                            'ALU/B,BMX/KMX,KMX/SC,SHF/ALU,QK/SHF''
;2915    Q_SHF                           'QK/SHF''
;2916
;2917    R(DST)_ALU                      'SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST''
;2918    R(DST)_D                        'RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST''
;2919    R(DST)_D.SXT[].RIGHT            'RAMX/D,AMX/RAMX.SXT,DT/a1,ALU/A,SHF/RIGHT,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST''
;2920
;2921    R(PRN)_0+D.RLOG                 'ALU/A+B.RLOG,BMX/RBMX,RBMX/D,AMX/RAMX.OXT,DT/LONG,R(PRN)_ALU''
;2922    R(PRN)_ALU                      'SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN''
;2923    R(PRN)_D                        'RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN''
;2924    R(PRN)_D+K[].RLOG               'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R(PRN)_ALU''
;2925    R(PRN)_D-K[].RLOG               'RAMX/D,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A-B.RLOG,DT/LONG,R(PRN)_ALU''
;2926    R(PRN)_D.OR.Q                   'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,R(PRN)_ALU''
;2927    R(PRN)_D[]Q                     'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/a1,R(PRN)_ALU''
;2928    R(PRN)_K[]                      'KMX/a1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN''
;2929    R(PRN)_LA+K[].RLOG              'AMX/LA,KMX/a1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R(PRN)_ALU''
;2930    R(PRN)_LA+Q                     'AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN''
;2931    R(PRN)_LA-K[].RLOG              'AMX/LA,KMX/a1,BMX/KMX,ALU/A-B.RLOG,DT/LONG,R(PRN)_ALU''
;2932    R(PRN)_LA[]MASK                 'AMX/LA,BMX/MASK,ALU/a1,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN''
;2933    R(PRN)_LC                       'BMX/LC,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN''
;2934    R(PRN)_PACK.FP                  'BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN''
;2935    R(PRN)_Q                        'RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN''
;2936    R(PRN)_Q+K[].RLOG               'RAMX/Q,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R(PRN)_ALU''
;2937    R(PRN)_Q-K[].RLOG               'RAMX/Q,AMX/RAMX,KMX/a1,BMX/KMX,ALU/A-B.RLOG,DT/LONG,R(PRN)_ALU''
;2938    R(PRN+1)_ALU                    'SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1''
;2939    R(PRN+1)_D                      'RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1''
;2940    R(PRN+1)_D.OR.Q                 'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1''
;2941    R(PRN+1)_K[]                    'KMX/a1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1''
;2942    R(PRN+1)_LA                     'AMX/LA,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1''
;2943    R(PRN+1)_LC                     'BMX/LC,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1''
```

```
;2944    R(PRN+1)_Q                      "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1"
;2945
;2946    R(SC)_ALU                       "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
;2947    R(SC)_D                         "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
;2948    R(SC)_K[]                       "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
;2949    R(SC)_LA                        "AMX/LA,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
;2950    R(SC)_LA+D                      "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
;2951    R(SC)_LA-D                      "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
;2952    R(SC)_LC                        "ALU_LC,R(SC)_ALU"
;2953    R(SC)_Q                         "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
;2954
;2955    R(SP1)_ALU                      "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
;2956    R(SP1)_D                        "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
;2957    R(SP1)_K[]                      "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
;2958    R(SP1)_PACK.FP                  "BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
;2959    R(SP1)_Q                        "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
;2960    R(SP1+1)_LC                     "BMX/LC,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1+1"
;2961    R(SP1+1)_Q                      "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1+1"
;2962
;2963    R(SRC!1)_ALU                    "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.OR.1"
;2964    R(SRC!1)_D(B)                   "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.OR.1"
;2965    R(SRC)_ALU                      "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
;2966    R(SRC)_D                        "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
;2967    R(SRC)_D(B)                     "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
;2968    R(SRC)_D+K[].RLOG               "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,DT/WORD,R(SRC)_ALU"
;2969    R(SRC)_D-K[].RLOG               "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B.RLOG,DT/WORD,R(SRC)_ALU"
;2970    R(SRC)_LC                       "BMX/LC,ALU/B,R(SRC)_ALU"
;2971    R(SRC)_Q                        "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
;2972
;2973    R6_D+K[].RLOG                   "SPO.R/WRITE.RAB,SPO.RAB/R6,RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,SHF/ALU"
;2974    R6_LA+K[].RLOG                  "AMX/LA,BMX/KMX,KMX/@1,ALU/A+B.RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/R6"
;2975    R6_LA-K[].RLOG                  "AMX/LA,BMX/KMX,KMX/@1,ALU/A-B.RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/R6"
;2976
;2977    RC(SC)_0-LC                     "ALU_0-LC,RC(SC)_ALU"
;2978    RC(SC)_ALU                      "SHF/ALU,SPO/WRITE.RC.SC"
;2979    RC(SC)_ALU.RIGHT                "SPO/WRITE.RC.SC,SHF/RIGHT"
;2980    RC(SC)_D                        "ALU_D,RC(SC)_ALU"
;2981    RC(SC)_Q                        "ALU_Q,RC(SC)_ALU"
;2982
;2983    RC[]&VA_D+Q                     "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2984    RC[]_0                          "AMX/RAMX.0XT,DT/LONG,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2985    RC[]_0+K[]+1                    "AMX/RAMX.0XT,DT/LONG,KMX/@2,BMX/KMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2986    RC[]_0+LC+1                     "AMX/RAMX.0XT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2987    RC[]_0+MASK+1                   "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2988    RC[]_0+MASK+1.RIGHT2            "AMX/RAMX.0XT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1"
;2989    RC[]_0-D                        "AMX/RAMX.0XT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2990    RC[]_ALU                        "SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2991    RC[]_ALU.LEFT                   "SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1"
;2992    RC[]_ALU.LEFT2                  "SPO.R/WRITE.RC,SPO.RC/@1,SHF/ALU.DT,DT/LONG"
;2993    RC[]_ALU.LEFT3                  "SPO.R/WRITE.RC,SPO.RC/@1,SHF/LEFT3"
;2994    RC[]_ALU.RIGHT                  "SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/@1"
;2995    RC[]_ALU.RIGHT2                 "SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1"
;2996    RC[]_D                          "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2997    RC[]_D(B)                       "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
;2998    RC[]_D+K[]                      "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
```

```
;2999   RC[]_D-K[]                    'RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3000   RC[]_D.OXT[]                  'RAMX/D,AMX/RAMX.OXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3001   RC[]_D.AND.K[]                'RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3002   RC[]_D.AND.MASK               'RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3003   RC[]_D.ANDNOT.Q               'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3004   RC[]_D.CTX                    'RAMX/D,AMX/RAMX,ALU/A,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1''
;3005   RC[]_D.LEFT                   'RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1''
;3006   RC[]_D.LEFT3                  'RAMX/D,AMX/RAMX,ALU/A,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/@1''
;3007   RC[]_D.OR.K[]                 'RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3008   RC[]_D.OR.Q                   'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3009   RC[]_D.ORNOT.K[]              'SPO.RC/@1,SPO.R/WRITE.RC,ALU/ORNOT,AMX/RAMX,RAMX/D,BMX/KMX,KMX/@2,SHF/ALU''
;3010   RC[]_D.SXT[]                  'RAMX/D,AMX/RAMX.SXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3011   RC[]_K[]                      'KMX/@2,BMX/KMX,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3012   RC[]_K[]+1                    'AMX/RAMX.OXT,DT/LONG,KMX/@2,BMX/KMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3013   RC[]_K[].LEFT2                'KMX/@2,BMX/KMX,ALU/B,SHF/ALU.DT,DT/LONG,SPO.R/WRITE.RC,SPO.RC/@1''
;3014   RC[]_K[].LEFT3                'KMX/@2,BMX/KMX,ALU/B,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/@1''
;3015   RC[]_K[].RIGHT2               'KMX/@2,BMX/KMX,ALU/B,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1''
;3016   RC[]_LA                       'AMX/LA,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3017   RC[]_LA+LB.CTX                'AMX/LA,BMX/LB,ALU/A+B,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1''
;3018   RC[]_LA-K[]                   'AMX/LA,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3019   RC[]_LA.AND.K[]               'ALU_LA.AND.K[@2],RC[@1]_ALU''
;3020   RC[]_LA.CTX                   'AMX/LA,ALU/A,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1''
;3021   RC[]_LB                       'BMX/LB,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3022   RC[]_LB.LEFT                  'BMX/LB,ALU/B,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1''
;3023   RC[]_LC                       'BMX/LC,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3024   RC[]_NOT.Q                    'RAMX/Q,AMX/RAMX,ALU/NOTA,RC[@1]_ALU''
;3025   RC[]_PACK.FP                  'BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3026   RC[]_PC                       'BMX/PC,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3027   RC[]_Q                        'RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3028   RC[]_Q+1                      'ALU_Q+Q+1,RC[@1]_ALU''
;3029   RC[]_Q+K[]                    'RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3030   RC[]_Q+LC                     'ALU/A+B,RAMX/Q,AMX/RAMX,BMX/LC,SPO.R/WRITE.RC,SPO.RC/@1''
;3031   RC[]_Q+PC                     'RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3032   RC[]_Q+PC+1                   'RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3033   RC[]_Q-K[]                    'RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3034   RC[]_Q-LC                     'ALU/A-B,RAMX/Q,AMX/RAMX,BMX/LC,SPO.R/WRITE.RC,SPO.RC/@1''
;3035   RC[]_Q-MASK-1                 'RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU.SPO.R/WRITE.RC,SPO.RC/@1''
;3036   RC[]_Q.OXT[]                  'RAMX/Q,AMX/RAMX.OXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3037   RC[]_Q.AND.K[]                'RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3038   RC[]_Q.ANDNOT.K[]             'RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3039   RC[]_Q.LEFT                   'RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1''
;3040   RC[]_Q.LEFT3                  'RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/@1''
;3041   RC[]_Q.RIGHT                  'RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/@1''
;3042   RC[]_Q.RIGHT2                 'ALU_Q,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/@1''
;3043   RC[]_Q.SXT[]                  'RAMX/Q,AMX/RAMX.SXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1''
;3044   RC[]_RLOG.RIGHT               'BMX/0,MSC/READ.RLOG,ALU/B,SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/@1''
;3045
;3046   R[]&VA_LA+K[]                 ''AMX/LA,KMX/@2,BMX/KMX,ALU/A+B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1''
;3047   R[]&VA_LA-K[]                 ''AMX/LA,KMX/@2,BMX/KMX,ALU/A-B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1''
;3048   R[]&VA_LA-K[].RLOG            ''AMX/LA,KMX/@2,BMX/KMX,ALU/A-B.RLOG,DT/LONG,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1''
;3049   R[]&VA_Q-K[]                  ''RAMX/Q,AMX/RAMX,KMX/@2,BMX/KMX,ALU/A-B,VAK/LOAD SPO.R/WRITE.RAB,SPO.RAB/@1''
;3050   R[]_0                         ''SPO.R/WRITE.RAB,SPO.RAB/@1,AMX/RAMX.OXT,DT/LONG,ALU/A,SHF/ALU''
;3051   R[]_0+LB+1                    ''AMX/RAMX.OXT DT/LONG,BMX/LB,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1''
;3052   R[]_0-1                       ''AMX/RAMX.OXT,DT/LONG,BMX/KMX,KMX/.1,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1''
;3053   R[]_0-D                       ''AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1''
```

```
;3054   R[]_0-K[]                    "AMX/RAMX.0XT,DT/LONG,KMX/a2,BMX/KMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3055   R[]_0-LB                     "AMX/RAMX.0XT,DT/LONG,BMX/LB,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3056   R[]_0-Q                      "AMX/RAMX.0XT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3057   R[]_ALU                      "SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3058   R[]_ALU.LEFT                 "SPO.R/WRITE.RAB,SPO.RAB/a1,SHF/LEFT"
;3059   R[]_ALU.LEFT3                "SPO.R/WRITE.RAB,SPO.RAB/a1,SHF/LEFT3"
;3060   R[]_ALU.RIGHT                "SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3061   R[]_ALU.RIGHT2               "SPO.R/WRITE.RAB,SPO.RAB/a1,SHF/RIGHT2"
;3062   R[]_D                        "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/D,AMX/RAMX,ALU/A,SHF/ALU"
;3063   R[]_D+K[]                    "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/D,AMX/RAMX,KMX/a2,BMX/KMX,ALU/A+B,SHF/ALU"
;3064   R[]_D+Q                      "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU"
;3065   R[]_D+Q+1                    "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1,SHF/ALU"
;3066   R[]_D-K[]                    "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/D,AMX/RAMX,KMX/a2,BMX/KMX,ALU/A-B,SHF/ALU"
;3067   R[]_D-LC-1                   "ALU_D-LC-1,R[a1]_ALU"
;3068   R[]_D-Q                      "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU"
;3069   R[]_D.AND.K[]                "SPO.R/WRITE.RAB,SPO.RAB/a1,ALU/AND,AMX/RAMX,RAMX/D,BMX/KMX,KMX/a2,SHF/ALU"
;3070   R[]_D.OR.LC                  "SPO.R/WRITE.RAB,SPO.RAB/a1,ALU/OR,AMX/RAMX,RAMX/D,BMX/LC,SHF/ALU"
;3071   R[]_D.OR.PACK.FP             "SPO.R/WRITE.RAB,SPO.RAB/a1,ALU/OR,AMX/RAMX,RAMX/D,BMX/PACKED.FL,SHF/ALU"
;3072   R[]_D.OR.Q                   "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU"
;3073   R[]_K[]                      "BMX/KMX,KMX/a2,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3074   R[]_LA                       "SPO.R/WRITE.RAB,SPO.RAB/a1,AMX/LA,ALU/A,SHF/ALU"
;3075   R[]_LA+D                     "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3076   R[]_LA+D+1                   "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3077   R[]_LA+K[]                   "AMX/LA,BMX/KMX,KMX/a2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3078   R[]_LA+K[]+1                 "AMX/LA,BMX/KMX,KMX/a2,ALU/A+B+1,R[a1]_ALU"
;3079   R[]_LA+K[].RLOG              "AMX/LA,BMX/KMX,KMX/a2,ALU/A+B.RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3080   R[]_LA+LC                    "AMX/LA,BMX/LC,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3081   R[]_LA+MASK+1                "AMX/LA,BMX/MASK,ALU/A+B+1,R[a1]_ALU"
;3082   R[]_LA+Q                     "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3083   R[]_LA-D                     "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3084   R[]_LA-K[]                   "AMX/LA,BMX/KMX,KMX/a2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3085   R[]_LA-K[].RLOG              "AMX/LA,BMX/KMX,KMX/a2,ALU/A-B.RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3086   R[]_LA-MASK-1                "ALU/A-B-1,AMX/LA,BMX/MASK,SPO.R/WRITE.RAB,SPO.RAB/a1,SHF/ALU"
;3087   R[]_LA-Q                     "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3088   R[]_LA.AND.K[]               "AMX/LA,BMX/KMX,KMX/a2,ALU/AND,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3089   R[]_LA.OR.D                  "AMX/LA,RBMX/D,BMX/RBMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3090   R[]_LA.ORNOT.MASK            "AMX/LA,BMX/MASK,ALU/ORNOT,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3091   R[]_LB                       "BMX/LB,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3092   R[]_LC                       "BMX/LC,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3093   R[]_LC.RIGHT                 "BMX/LC,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3094   R[]_NOT.0                    "AMX/RAMX.0XT,DT/LONG,ALU/NOTA,R[a1]_ALU"
;3095   R[]_NOT.D                    "RAMX/D,AMX/RAMX,ALU/NOTA,R[a1]_ALU"
;3096   R[]_NOT.MASK                 "BMX/MASK,AMX/RAMX.0XT,DT/LONG,ALU/ORNOT,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3097   R[]_NOT.Q                    "RAMX/Q,AMX/RAMX,ALU/NOTA,R[a1]_ALU"
;3098   R[]_PACK.FP                  "BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
;3099   R[]_Q                        "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU"
;3100   R[]_Q+1                      "ALU_Q+Q+1,R[a1]_ALU"
;3101   R[]_Q+5                      "SPO.R/WRITE.RAB,SPO.RAB/a1,ALU/A+B+1,BMX/KMX,KMX/.4,AMX/RAMX,RAMX/Q,SHF/ALU"
;3102   R[]_Q+K[]                    "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/Q,AMX/RAMX,BMX/KMX,KMX/a2,ALU/A+B,SHF/ALU"
;3103   R[]_Q+LB                     "SPO.R/WRITE.RAB,SPO.RAB/a1,ALU/A+B,AMX/RAMX,BMX/LB,RAMX/Q,SHF/ALU"
;3104   R[]_Q+LC                     "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU"
;3105   R[]_Q-D                      "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU"
;3106   R[]_Q-D-1                    "SPO.R/WRITE.RAB,SPO.RAB/a1,ALU/A-B-1,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D,SHF/ALU"
;3107   R[]_Q-K[]                    "SPO.R/WRITE.RAB,SPO.RAB/a1,RAMX/Q,AMX/RAMX,BMX/KMX,KMX/a2,ALU/A-B,SHF/ALU"
;3108   R[]_Q-K[].RLOG               "RAMX/Q,AMX/RAMX,BMX/KMX,KMX/a2,ALU/A-B.RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/a1"
```

```
;3109   R[]_Q-LC                          'SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU''
;3110   R[]_Q.AND.K[]                     'ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/@1,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@2''
;3111   R[]_Q.ANDNOT.K[]                  'SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@2,SHF/ALU''
;3112   R[]_Q.OR.D                        'SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/OR,AMX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D,SHF/ALU''
;3113   R[]_Q.ORNOT.K[]                   'SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/ORNOT,SHF/ALU''
;3114   R[]_Q.RIGHT.1                     'ALU_Q,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1''
;3115   R[]_RLOG.RIGHT.1                  'BMX70,MSC/READ.RLOG,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1''
;3116
;3117   SC&STATE_STATE-R[](EXP)           'LAB_R[@1],AMX/LA,EBMX/AMX.EXP,MSC/LOAD.STATE,EALU/A-B,SMX/EALU,SCK/LOAD''
;3118   SC_0(A)                           'AMX7/RAMX.0XT,DT/LONG,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD''
;3119   SC_0-K[]                          'BMX/KMX,KMX/@1,AMX/RAMX.0XT,DT/LONG,ALU/A-B,SMX/ALU,SCK/LOAD''
;3120   SC_ALU                            'SMX/ALU,SCK/LOAD''
;3121   SC_ALU(EXP)                       'SMX/ALU.EXP,SCK/LOAD''
;3122   SC_D                              'RAMX/D,AMX/RAMX,ALU/A,SMX/ALU,SCK/LOAD''
;3123   SC_D(EXP)                         'RAMX/D,AMX/RAMX,ALU/A,SMX/ALU.EXP,SCK/LOAD''
;3124   SC_D(EXP)(A)                      'RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD''
;3125   SC_D(EXP)(B)                      'RBMX/D,BMX/RBMX,ALU/B,SMX/ALU.EXP,SCK/LOAD''
;3126   SC_D-K[]                          'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SMX/ALU,SCK/LOAD''
;3127   SC_D.0XT[]-K[]                    'RAMX/D,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B,SMX/ALU,SCK/LOAD''
;3128   SC_D.0XT[].XOR.K[]                'RAMX/D,AMX/RAMX.0XT,DT/@1,BMX/KMX,KMX/@2,ALU/XOR,SC_ALU''
;3129   SC_D.AND.K[]                      'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD''
;3130   SC_D.OR.K[]                       'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SMX/ALU,SCK/LOAD''
;3131   SC_D.SXT[]                        'RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SMX/ALU,SCK/LOAD''
;3132   SC_EALU                           'SMX/EALU,SCK/LOAD''
;3133   SC_FE                             'SMX/FE,SCK/LOAD''
;3134   SC_K[]                            'KMX/@1,EBMX/KMX,EALU/B,SMX/EALU,SCK/LOAD''
;3135   SC_K[].ALU                        'KMX/@1,BMX/KMX,ALU/B,SMX/ALU,SCK/LOAD''
;3136   SC_LA                             'AMX/LA,ALU/A,SMX/ALU,SCK/LOAD''
;3137   SC_LA.AND.K[]                     'AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD''
;3138   SC_LC(EXP)                        'BMX/LC,ALU/B,SMX/ALU.EXP,SCK/LOAD''
;3139   SC_NABS(SC-FE)                    'EBMX/FE,EALU/NABS.A-B,SMX/EALU,SCK/LOAD''
;3140   SC_PSLADDR                        'SMX/EALU,EBMX/KMX,SCK/LOAD,KMX/.F,EALU/B''
;3141   SC_Q                              'RAMX/Q,AMX/RAMX,ALU/A,SMX/ALU,SCK/LOAD''
;3142   SC_Q(EXP)                         'RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD''
;3143   SC_Q(EXP)(B)                      'RBMX/Q,BMX/RBMX,ALU/B,SMX/ALU.EXP,SCK/LOAD''
;3144   SC_Q+K[]                          'RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/A+B,SMX/ALU,SCK/LOAD''
;3145   SC_Q-K[]                          'RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/A-B,SMX/ALU,SCK/LOAD''
;3146   SC_Q.AND.K[]                      'RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/AND,SMX/ALU,SCK/LOAD''
;3147   SC_Q.OR.K[]                       'RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@1,ALU/OR,SMX/ALU,SCK/LOAD''
;3148   SC_Q.SXT[]                        'RAMX/Q,AMX/RAMX.SXT,DT/@1,ALU/A,SMX/ALU,SCK/LOAD''
;3149   SC_RC[]                           'SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SMX/ALU,SCK/LOAD''
;3150   SC_RC[](EXP)                      'SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SMX/ALU.EXP,SCK/LOAD''
;3151   SC_R[]                            'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SMX/ALU,SCK/LOAD''
;3152   SC_R[](EXP)                       'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SMX/ALU.EXP,SCK/LOAD''
;3153   SC_R[].AND.K[]                    'ALU/AND,AMX/LA,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/KMX,KMX/@2,SMX/ALU,SCK/LOAD''
;3154   SC_SC+1                           'EALU/A+1,SMX/EALU,SCK/LOAD''
;3155   SC_SC+EXP(Q)(A)                   'EALU/A+B,EBMX/AMX.EXP,SMX/EALU,SCK/LOAD,AMX/RAMX,RAMX/Q''
;3156   SC_SC+FE                          'EBMX/FE,EALU/A+B,SMX/EALU,SCK/LOAD''
;3157   SC_SC+K[]                         'KMX/@1,EBMX/KMX,EALU/A+B,SMX/EALU,SCK/LOAD''
;3158   SC_SC+SHF.VAL                     'EALU/A+B,EBMX/SHF.VAL,SMX/EALU,SCK/LOAD''
;3159   SC_SC-FE                          'EBMX/FE,EALU/A-B,SMX/EALU,SCK/LOAD''
;3160   SC_SC-K[]                         'KMX/@1,EBMX/KMX,EALU/A-B,SMX/EALU,SCK/LOAD''
;3161   SC_SC-SHF.VAL                     'EBMX/SHF.VAL,EALU/A-B,SMX/EALU,SCK/LOAD''
;3162   SC_SC.ANDNOT.FE                   'EBMX/FE,EALU/ANDNOT,SMX/EALU,SCK/LOAD''
;3163   SC_SC.ANDNOT.K[]                  'KMX/@1,EBMX/KMX,EALU/ANDNOT,SMX/EALU,SCK/LOAD''
```

```
;3164   SC_SC.OR.K[]                    'KMX/@1,EBMX/KMX,EALU/OR,SMX/EALU,SCK/LOAD''
;3165   SC_SHF.VAL                      'EBMX/SHF.VAL,EALU/B,SMX/EALU,SCK/LOAD''
;3166   SC_STATE                        'EALU/A,MSC/LOAD.STATE,SMX/EALU,SCK/LOAD''
;3167   SC_STATE.ANDNOT.K[]             'EALU/ANDNOT,EBMX/KMX,MSC/LOAD.STATE,SMX/EALU,SCK/LOAD,KMX/@1''
;3168   SC_STATE.OR.K[]                 'EALU/OR,EBMX/KMX,MSC/LOAD.STATE,SMX/EALU,SCK/LOAD,KMX/@1''
;3169   SD_NOT.SD                       'SGN/NOT.SD''
;3170   SD_SS                           'SGN/SD.FROM.SS''
;3171   SS_0&SD_0                       'SGN/CLR.SD+SS''
;3172   SS_ALU15                        'SGN/LOAD.SS''
;3173   SS_SD                           'SGN/SS.FROM.SD''
;3174   SS_SS.XOR.ALU15&SD_ALU15        'SGN/SS.XOR.ALU''
;3175   STATE_0(A)                      'AMX/RAMX.0XT,DT/LONG,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE''
;3176   STATE_AMX.EXP                   'EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE''
;3177   STATE_D(EXP)                    'RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE''
;3178   STATE_FE                        'EBMX/FE,EALU/B,MSC/LOAD.STATE''
;3179   STATE_FIRST                     'STATE_K[ZERO]''                    ;EDITPC STATES
;3180   STATE_INNEROBJ                  'STATE_K[.1]''                      ;MATCHC STATES
;3181   STATE_INNERSRC                  'STATE_K[.3]''
;3182   STATE_K[]                       'KMX/@1,EBMX/KMX,EALU/B,MSC/LOAD.STATE''
;3183   STATE_OUTER                     'STATE_K[ZERO]''
;3184   STATE_PREDEC                    'STATE_K[.80]''
;3185   STATE_Q(EXP)                    'RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE''
;3186   STATE_SC.VIA.KMX                'MSC/LOAD.STATE,EALU/B,EBMX/KMX,KMX/SC''
;3187   STATE_SKPLONG                   'STATE_K[.4]''                      ;SKPC STATES
;3188   STATE_STATE+1                   'EALU/A+1,MSC/LOAD.STATE''
;3189   STATE_STATE+FE                  'EBMX/FE,EALU/A+B,MSC/LOAD.STATE''
;3190   STATE_STATE+K[]                 'KMX/@1,EBMX/KMX,EALU/A+B,MSC/LOAD.STATE''
;3191   STATE_STATE-FE                  'EBMX/FE,EALU/A-B,MSC/LOAD.STATE''
;3192   STATE_STATE-K[]                 'KMX/@1,EBMX/KMX,EALU/A-B,MSC/LOAD.STATE''
;3193   STATE_STATE.AN.SKPLONG          'STATE_STATE.ANDNOT.K[.4]''
;3194   STATE_STATE.AN.5TO0             'STATE_STATE.ANDNOT.K[.3F]''
;3195   STATE_STATE.AN.6TO4             'STATE_STATE.ANDNOT.K[.7F]''
;3196   STATE_STATE.AN.DESTDBL          'STATE_STATE.ANDNOT.K[.6]''
;3197   STATE_STATE.AN.NOTPREDEC        'STATE_STATE.ANDNOT.K[.7F]''
;3198   STATE_STATE.AN.PREDECZERO       'STATE_STATE.ANDNOT.K[.C0]''
;3199   STATE_STATE.ANDNOT.FE           'EBMX/FE,EALU/ANDNOT,MSC/LOAD.STATE''
;3200   STATE_STATE.ANDNOT.K[]          'KMX/@1,EBMX/KMX,EALU/ANDNOT,MSC/LOAD.STATE''
;3201   STATE_STATE.ANDNOT.SHF.VAL      'MSC/LOAD.STATE,EBMX/SHF.VAL,EALU/ANDNOT''
;3202   STATE_STATE.OR.FE               'EALU/OR,EBMX/FE,MSC/LOAD.STATE''
;3203   STATE_STATE.OR.K[]              'KMX/@1,EBMX/KMX,EALU/OR,MSC/LOAD.STATE''
;3204   STATE_STATE.OR.ADJINP           'STATE_STATE.OR.K[.3]''
;3205   STATE_STATE.OR.DEST             'STATE_STATE.OR.K[.4]''
;3206   STATE_STATE.OR.DESTDBL          'STATE_STATE.OR.K[.6]''
;3207   STATE_STATE.OR.FILL             'STATE_STATE.OR.K[.7]''
;3208   STATE_STATE.OR.FLOAT            'STATE_STATE.OR.K[.60]''
;3209   STATE_STATE.OR.MOVE             'STATE_STATE.OR.K[.50]''
;3210   STATE_STATE.OR.PATT1            'STATE_STATE.OR.K[.1]''
;3211   STATE_STATE.OR.PATT2            'STATE_STATE.OR.K[.2]''
;3212   SWAPC                           'DK/BYTE.SWAP''
;3213
;3214   VA_ALU                          'VAK/LOAD''
;3215   VA_D                            'RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD''
;3216   VA_D+K[]                        'RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD''
;3217   VA_D+LC                         'RAMX/D,AMX/RAMX,BMX/LC,ALU'A+B,VAK/LOAD''
;3218   VA_D+Q                          'RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD''
```

```
;3219   VA_D.OXT[]+Q                    'RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/RBMX,ALU/A+B,VAK/LOAD''
;3220   VA_D.ANDNOT.K[]                 'RAMX/D,AMX/RAMX,BMX/KMX,KMX/@1,ALU/ANDNOT,VAK/LOAD''
;3221   VA_K[]                          'KMX/@1,BMX/KMX,ALU/B,VAK/LOAD''
;3222   VA_LA                           'AMX/LA,ALU/A,VAK/LOAD''
;3223   VA_LA+D                         'AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,VAK/LOAD''
;3224   VA_LA+K[]                       'AMX/LA,BMX/KMX,KMX/@1,ALU/A+B,VAK/LOAD''
;3225   VA_LA+K[]+1                     'AMX/LA,BMX/KMX,KMX/@1,ALU/A+B+1,VAK/LOAD''
;3226   VA_LA+PC                        'AMX/LA,BMX/PC,ALU/A+B,VAK/LOAD''
;3227   VA_LA+Q                         'AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD''
;3228   VA_LA-D                         'AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,VAK/LOAD''
;3229   VA_LA-K[]                       'AMX/LA,BMX/KMX,KMX/@1,ALU/A-B,VAK/LOAD''
;3230   VA_LA-K[]-1                     'AMX/LA,BMX/KMX,KMX/@1,ALU/A-B-1,VAK/LOAD''
;3231   VA_LA-Q                         'VAK/LOAD,ALU/A-B,AMX/LA,BMX/RBMX,RBMX/Q,SHF/ALU''
;3232   VA_LA.AND.LC                    'AMX/LA,BMX/LC,ALU/AND,VAK/LOAD''
;3233   VA_LA.ANDNOT.K[]                'AMX/LA,BMX/KMX,KMX/@1,ALU/ANDNOT,VAK/LOAD''
;3234   VA_LB+D.OXT                     'BMX/LB,ALU/A+B,AMX/RAMX.OXT,DT/BYTE,VAK/LOAD''
;3235   VA_PC                           'BMX/PC,ALU/B,VAK/LOAD''
;3236   VA_Q                            'RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD''
;3237   VA_Q+D                          'VAK/LOAD,ALU/A+B,AMX/RAMX,BMX/RBMX,RAMX/Q,RBMX/D,SHF/ALU''
;3238   VA_Q+K[]                        'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD''
;3239   VA_Q+LB                         'RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B,VAK/LOAD''
;3240   VA_Q+LB.PC                      'RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD''
;3241   VA_Q+LC                         'RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD''
;3242   VA_Q+PC                         'RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD''
;3243   VA_Q-K[]                        'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD''
;3244   VA_Q-LB                         'RAMX/Q,AMX/RAMX,BMX/LB,ALU/A-B,VAK/LOAD''
;3245   VA_Q.ANDNOT.K[]                 'RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,VAK/LOAD''
;3246   VA_RC[]                         'SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,VAK/LOAD''
;3247   VA_R[]                          'SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,VAK/LOAD''
;3248   VA_VA+4                         'PCK/VA+4''
```

```
;3249    .TOC    ''      Macro definition      : Non-transfer macros''
;3250
;3251    B.FORK                          'LAB_R(SP1),QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/B.FORK''
;3252    BYTE                            'DT/BYTE''
;3253
;3254    C.FORK                          'SUB/SPEC,J/C.FORK''
;3255    CACHE.INVALIDATE                'MCT/INVALIDATE,VAK/NOP''
;3256    CALL                            'SUB/CALL''
;3257    CALL[]                          'CALL,J/@1''
;3258    CHK.FLT.OPR                     'MSC/CHK.FLT.OPR''
;3259    CHK.ODD.ADDR                    'MSC/CHK.ODD.ADDR''
;3260    CLK.UBCC                        'CCK/LOAD.UBCC''
;3261
;3262    CLR.FPD                         'MSC/CLR.FPD''
;3263    CLR.IB.COND                     ''IBC/CLR.1-5.COND''
;3264    CLR.IB.OPC                      ''IBC/CLR.0,IEK/ISTR''
;3265    CLR.IB.SPEC                     ''IBC/CLR.1''
;3266    CLR.IB0-1                       ''IBC/CLR.0.1,IEK/ISTR''
;3267    CLR.IB0-3                       ''IBC/CLR.0-3''             ;DISCARD -11 INSTR & OPERAND
;3268    CLR.IB2-3                       ''IBC/CLR.2.3''             ;11 MODE DISCARD ISTREAM OPERAND
;3269    CLR.IB2-5                       ''IBC/CLR.1-5.COND''        ;2ND PART OF Q/D IMMEDIATE
;3270    CLR.NEST.ERR                    'MSC/CLR.NEST.ERR''
;3271    CLR.SD&SS                       'SGN/CLR.SD+SS''
;3272
;3273    E.FORK                          'SUB/SPEC,J/E.FORK''
;3274    EXCEPT.ACK                      ''IEK/EACK''
;3275
;3276    FLUSH.IB                        ''IBC/FLUSH,VAK/LOAD,IEK/ISTR''
;3277
;3278    G.FORK                          'SUB/SPEC,J/G.FORK''
;3279
;3280    INHIBIT.IB                      'MCT/MEM.NOP''
;3281    INTRPT.ACK                      ''IEK/IACK''
;3282    INTRPT.STROBE                   ''IEK/ISTR''
;3283    IRD                             ''IRD0,CLK.UBCC,IRD1,SUB/SPEC,J/A.FORK''
;3284    IRD.11                          'LA_R(DST)&LB_R(SRC),D_LB.PC,VAK/LOAD,Q_IB.DATA,SC_K[.10],PCK/PC+N,MSC/IRD,SUB/SPEC,J/DPO''
;3285    IRD0                            'LA_R(SP2)&LB_R(SP1),D&VA_LB,SC_ALU(EXP),FE_LA(EXP),SS_ALU15''
;3286    IRD1                            'MSC/IRD,QK/ID,MCT/ALLOW.IB.READ,IBC/CLR.1-5.COND,PCK/PC+N''
;3287
;3288    LOAD.ACC.CC                     'MSC/LOAD.ACC.CC''
;3289    LOAD.IB                         'VAK/NOP,MCT/READ.V.NEWPC''
;3290    LOAD.IB.11                      'VAK/NOP,MCT/READ.V.NEWPC''
;3291    LONG                            'DT/LONG''
;3292
;3293    MEMORY.NOP                      'MCT/MEM.NOP''
;3294    MUL.0XT                         'SI/MUL+,SC_SC-K[.1],BEN/MUL''
;3295    MUL.1XT                         'SI/MUL-,SC_SC-K[.1],BEN/MUL''
;3296    MULM.DONE                       'D_D.RIGHT2,SI/MUL-,INTRPT.STROBE''
;3297    MULP.DONE                       'D_D.RIGHT2,SI/MUL+,INTRPT.STROBE''
;3298
;3299    POLY.DONE                       ''ACF/CONTROL,ACM/POLY.DONE''
;3300
;3301    RETURN0                         'SUB/RET,J/0''
;3302    RETURN1                         'SUB/RET,J/1''
;3303    RETURN10                        'SUB/RET,J/10''
```

```
;3304    RETURN100                       "SUB/RET,J/100"
;3305    RETURN10C                       "SUB/RET,J/10C"
;3306    RETURN10E                       "SUB/RET,J/10E"
;3307    RETURN12                        "SUB/RET,J/12"
;3308    RETURN18                        "SUB/RET,J/18"
;3309    RETURN1F                        "SUB/RET,J/1F"
;3310    RETURN2                         "SUB/RET,J/2"
;3311    RETURN20                        "SUB/RET,J/20"
;3312    RETURN24                        "SUB/RET,J/24"
;3313    RETURN3                         "SUB/RET,J/3"
;3314    RETURN4                         "SUB/RET,J/4"
;3315    RETURN40                        "SUB/RET,J/40"
;3316    RETURN60                        "SUB/RET,J/60"
;3317    RETURN61                        "SUB/RET,J/61"
;3318    RETURN8                         "SUB/RET,J/8"
;3319    RETURN9                         "SUB/RET,J/9"
;3320    RETURNF                         "SUB/RET,J/0F"
;3321    RETURN[]                        "SUB/RET,J/@1"
;3322
;3323    SET.CC(BYTE)                    "CCK/INST.DEP,DT/BYTE"
;3324    SET.CC(INST)                    "CCK/INST.DEP,DT/INST.DEP"
;3325    SET.CC(LONG)                    "CCK/INST.DEP,DT/LONG"
;3326    SET.CC(ROR)                     "CCK/ROR"
;3327    SET.CC(WORD)                    "CCK/INST.DEP,DT/WORD"
;3328    SET.FPD                         "MSC/SET.FPD"
;3329    SET.NEST.ERR                    "MSC/SET.NEST.ERR"
;3330    SET.PSL.C(AMX)                  "CCK/C_AMX0"
;3331    SET.V                           "CCK/SET.V"
;3332    SPEC                            "LAB_R(SP1),Q_IB.DATA,CLR.IB.COND,PC_PC+N,MCT/ALLOW.IB.READ,SUB/SPEC,J/C.FORK"
;3333    SPECG                           "LAB_R(SP1),Q_IB.DATA,CLR.IB.COND,PC_PC+N,MCT/ALLOW.IB.READ,SUB/SPEC,J/G.FORK"
;3334    START.IB                        "IBC/START"
;3335    STOP.IB                         "IBC/STOP"
;3336
;3337    TEST.TB.RCHK                    "MCT/TEST.RCHK,VAK/NOP"
;3338    TEST.TB.WCHK                    "MCT/TEST.WCHK,VAK/NOP"
;3339    TRAP.ACC[]                      "ACF/TRAP,ACM/@1"
;3340
;3341    WORD                            "DT/WORD"
;3342    WRITE.DEST                      "LAB_R(SP1),QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/WRD"
;3343    WRITE.G.DEST                    "LAB_R(SP1),QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/WRG"
```

```
;3344   .TOC     ''      Macro definition      : Branch enable macrcs''
;3345
;3346   AC.LOW?                          'BEN/INTERRUPT'' ;,J3/3''
;3347   ACC.SYNC?                        'BEN/ACCEL''     ;,J3/3''
;3348   ACCEL?                           'BEN/ACCEL''
;3349   ALIGNED?                         'BEN/TB.TEST''   ;,J5/17''
;3350   ALU.N?                           'BEN/ALU''       ;,J4/07''
;3351   ALU1-0?                          'BEN/ALU1-0''
;3352   ALU?                             'BEN/ALU''
;3353
;3354   BCDSGN?                          'BEN/DECIMAL''   ;,J2/2''
;3355
;3356   C31?                             'BEN/C31''
;3357   CONSOLE.MODE?                    'BEN/PSL.MODE''  ;,J5/1B''
;3358
;3359   D(1)?                            'BEN/MUL''
;3360   D.B0?                            'BEN/D.BYTES''   ;,J4/0E''
;3361   D.B1?                            'BEN/D.BYTES''   ;,J4/0D''
;3362   D.B2?                            'BEN/D.BYTES''   ;,J4/08''
;3363   D.BYTES?                         'BEN/D.BYTES''
;3364   D.NE.0?                          'BEN/SIGNS''     ;,J3/5'' ;PREFERED FORM
;3365   D0?                              'BEN/D3-0''      ;,J4/0E''
;3366   D2-0?                            'BEN/D3-0''      ;,J4/08''
;3367   D2?                              'BEN/D3-0''      ;,J4/08''
;3368   D3-0?                            'BEN/D3-0''
;3369   D31?                             'BEN/SIGNS''     ;,J3/6''
;3370   D3?                              'BEN/D3-0''      ;,J4/07''
;3371   DATA.TYPE?                       'BEN/DATA.TYPE''
;3372   DBL?                             'BEN/DATA.TYPE''
;3373
;3374   EALU.N?                          'BEN/EALU''      ;,J4/07''
;3375   EALU.Z?                          'BEN/EALU''      ;,J4/08''
;3376   EALU?                            'BEN/EALU''
;3377   END.DP1?                         'BEN/END.DP1''
;3378
;3379   FPD?                             'BEN/LAST.REF''  ;,J4/07''
;3380
;3381   IB.TEST?                         'BEN/IB.TEST''
;3382   INT?                             'BEN/INTERRUPT''
;3383   INTERRUPT.REQ?                   'BEN/INTERRUPT'' ;,J3/5''
;3384   IR0.C31?                         'BEN/ALU''
;3385   IR0?                             'BEN/ALU''       ;,J4/0D''
;3386   IR1?                             'BEN/IR2-1''     ;,J3/6''
;3387   IR2-1?                           'BEN/IR2-1''
;3388
;3389   LAST.REF?                        'BEN/LAST.REF''
;3390
;3391   MODE.LSS.ASTLVL?                 'BEN/REI''       ;,J3/3''
;3392   MUL?                             'BEN/MUL''
;3393
;3394   NEST.ERR?                        'BEN/LAST.REF''  ;,J4/08''
;3395
;3396   PC.MODES?                        'BEN/PC.MODES''
;3397   PSL.C?                           'BEN/PSL.CC''    ;,J4/0E''
;3398   PSL.CC?                          'BEN/PSL.CC''
```

```
;3399    PSL.MODE?                       'BEN/PSL.MODE''
;3400    PSL.N?                          'BEN/PSL.CC''       ;,J4/7''
;3401    PSL.V?                          'BEN/PSL.CC''       ;,J4/0D''
;3402    PSL.Z?                          'BEN/PSL.CC''       ;,J4/0B''
;3403    PTE.VALID?                      'BEN/TB.TEST''      ;,J5/0F''
;3404
;3405    Q31?                            'BEN/SIGNS''        ;,J3/3''
;3406    QUAD?                           'BEN/DATA.TYPE''
;3407
;3408    RLOG.EMPTY?                     'BEN/ALU1-0''       ;,J4/7''
;3409    ROR?                            'BEN/ROR''
;3410
;3411    SC.GT.0?                        'BEN/SC''
;3412    SC.NE.0?                        'BEN/MUL''          ;,J3/3''
;3413    SC?                             'BEN/SC''
;3414    SIGNS?                          'BEN/SIGNS''
;3415    SRC.PC?                         'BEN/SRC.PC''       ;COMP MODE, BEN ON SRC R = PC
;3416    SS?                             'BEN/EALU''         ;,J4/0E''
;3417    STATE(7)?                       'STATE7-4?''
;3418    STATE0?                         'BEN/STATE3-0''     ;,J4/0E''
;3419    STATE1-0?                       'BEN/STATE3-0''     ;,J4/0C''
;3420    STATE1?                         'BEN/STATE3-0''     ;,J4/0D''
;3421    STATE2?                         'BEN/STATE3-0''     ;,J4/0B''
;3422    STATE3-0?                       'BEN/STATE3-0''
;3423    STATE3?                         'BEN/STATE3-0''     ;,J4/07''
;3424    STATE4?                         'BEN/STATE7-4''
;3425    STATE5?                         'BEN/STATE7-4''
;3426    STATE6?                         'BEN/STATE7-4''
;3427    STATE7-4?                       'BEN/STATE7-4''
;3428
;3429    TB.TEST?                        'BEN/TB.TEST''
;3430
;3431    VA31-30?                        'BEN/PSL.MODE''     ;,J5/07''
;3432    VA31?                           'BEN/PSL.MODE''     ;,J5/0F''
;3433
;3434    Z?                              'BEN/Z''
;3435    ZONED?                          'BEN/DECIMAL''      ;,J2/1''
;3436
                                         ;3437   .BIN                ;MAKE LISTING ROOM FOR BINARY FROM HERE ON
```

3437; This page intentionally left blank.

```
                                        ;3438   .TOC    'PATCH.MIC''
                                        ;3439   .TOC    'Revision 3.2''
                                        ;3440   ;       R. J. Avarbock, P. R. Guilbault
                                        ;3441
;3442   .NOSIN
;3443   .REGION/<WCSR1L>,<WCSR1H>/<WCSR2L>,<WCSR2H>
;3444
;3445   .TOC    ''      Revision History''
;3446
;3447
;3448   ; 03    Fix ACBD with negative ADDEND branch is on GTR instead of GEQ. No FPLA
;3449   ;       Patch no. 075, PLA trap PCS053F to WCS115F. Fix POLYF rounding
;3450   ;       Patch no. 076, PLA trap PCS055F to WCS1193. Fix POLYD rounding
;3451   ;       Patch no. 077, PLA trap PCS0594 to WCS1195. Fix ASHP overflow.
;3452   ;       Correct Patch 59 to fix ASHP setting PSL<v> on negative shift count
;3453   ;       Patch no. 078, PLA trap PCS0D61 to WCS117E. Fix MTPR SCBB
;3454   ;       Patch no. 080, PLA trap PCS014C to WCS1147. Floating faults.
;3455   ;                      PLA trap PCS014E to WCS1147. Floating faults.
;3456   ;                      PLA trap PCS017E to WCS1147. Floating faults.
;3457   ;       Patch no. 081, PLA trap PCS013E to WCS114F. Floating faults.
;3458   ;       Patch no. 082, PLA trap PCS01DC to WCS1150. Floating faults.
;3459   ;                      PLA trap PCS01DE to WCS1150. Floating faults.
;3460   ;                      PLA trap PCS043C to WCS1150. Floating faults.
;3461   ;       Patch no. 085, PLA trap PCS0E00 to WCS1151. Floating faults.
;3462   ;       Patch no. 083, PLA trap PCS03E4 to WCS117F. Floating faults.
;3463   ;       Patch no. 084, PLA trap PCS05BE to WCS1194. Floating faults.
;3464   ;       Patch no. 079, PLA trap PCS0BD3 to WCS1196. Fix ADDP/SUBP 4&6 sign prob.
;3465   ;       Patch no. 086, PLA trap PCS0A62 to WCS119F. EDIT FPD unpack.
;3466   ;       Patch no. 087, PLA trap PCS067E to WCS1197. Floating faults. Patch 022 no longer required.
;3467   ;       Patch no. 088, PLA trap PCS006B to WCS1198. Floating faults.
;3468   ;       Patch no. 089, PLA trap PCS07FA to WCS1199. (ADD,SUB)P(4,6) v bit.
;3469   ;       Modify Patch 025 to fix (ADD,SUB)P4 v bit.
;3470   ;       Modify Patch 069 to fix (ADD,SUB)P6 v bit.
;3471   ;       Patch no. 090, PLA trap PCS0BB2 to WCS119A. CVTTP int. problem.
;3472   ;       Patch no. 091, PLA trap PCS0D06 to WCS119B. PROBEx C bit.
;3473   ;       Patch no. 092, PLA trap PCS074E to WCS119C. CVTPT V bit.
;3474   ;       Patch no. 093, PLA trap PCS0C47 to WCS119D. (ADD,SUB)P(4,6) v bit.
;3475   ;       Add fix to patch 015 for G & H and change out 1180
;3476   ;       Change Patch 001 to fix ACB(D,G) when either INDEX or ADDEND is zero.
;3477   ;       Change macro that deal with condition codes.
;3478   ;       Correct Patch 80 to fix warm floating reg to reg overflow fault problem
;3479   ;       Modify patch 089 to fix similar cases to patch 089.
;3480   ;       Add 2nd FPLA trap for G&H. We now have 1 for G&H and 1 for no G&H
;3481   ;       Patch no. 094, PLA trap PCS0EDB trapped to WCS11A4.  REI CM PC problem.
;3482   ;       Patch no. 095, PLA trap PCS06E4 to WCS119E.  MOVC(3,5) optimization.
;3483   ;       Patch no. 096, PLA trap PCS0564 trapped to WCS1159.  Add MTPR #3F.
;3484   ;       Patch no. 097, PLA trap PCS0B0D trapped to WCS11A5.  EDITPC restart.
;3485   ;       Patch no. 098, PLA trap PCS0B1D trapped to WCS11A6.  EDITPC restart.
;3486   ;       Modify patch 094 to report offending odd address instead of offending REI.
;3487   ; 02    Change secondary WCS version number for 123
;3488   ; 01    Patch no. 074, PLA trap PCS0D47 to WCS1192. Fix MULP
;3489   ;       Remove PLA trap PCS040B to WCS117E. Not required add never implemented.
;3490   ;       Remove PLA trap PCS040F to WCS117F. Not required add never implemented.
;3491   ;       Remove PLA trap PCS03CF to WCS1150. Not required add never implemented.
;3492   ;       Remove PLA trap PCS0284 to WCS1151. Not required add never implemented.
```

```
;3493    ; 00    Standardize documentation of file and edit as required.
;3494    ;       Patch no. 073, PLA trap PCS0287 to WCS1181. Used to be in Q1.MIC
;3495    ; --    Start of history
;3496
                                        ;3497   .BIN
```

```
                                ;3498   .TOC "  PCS microcode patches : Version numbers"
                                ;3499
                                ;3500   .TITLE  "VAX11/780 Microcode : PCS 01, FPLA 0E, WCS124"
                                ;3501
                                ;3502
                                ;3503
                                ;3504
                                ;3505
                                ;3506
                                ;3507
                                ;3508
                                ;3509   ;F P L A    V E R S I O N
                                ;3510   ;The following location is used by the console to obtain the FPLA version no.
                                ;3511
                                ;3512   0F80:    ;-------------------------------; FPLA's map this loc to their version
U 0F80, 0000,003C,0180,F800,0000,0F80   ;3513           J/<.>                           ; number. The 6 lsb's are used.
                                ;3514
                                ;3515
                                ;3516
                                ;3517
                                ;3518
                                ;3519   ;P C S    V E R S I O N
                                ;3520   ;The following location is used by the console to obtain the PCS version no.
                                ;3521
                                ;3522   0111:    ;-------------------------------; Loop with version number as ID address
U 0111, 0000,003C,8580,2C00,0000,0111   ;3523           CID/READ.KMX,ID.ADDR/021,J/<.>  ; The 2 lsb's are used.
                                ;3524
                                ;3525
                                ;3526
                                ;3527
                                ;3528
                                ;3529   ;P R I M A R Y    W C S    V E R S I O N
                                ;3530   ;The primary WCS version number indicates which FPLA version is correct for
                                ;3531   ;this version of WCS. The console compares the primary WCS version with the
                                ;3532   ;FPLA version and reports a mismatch warning if they are different.
                                ;3533
                                ;3534   1111:    ;-------------------------------; Loop with version number as ID address
U 1111, 0000,003C,3980,2C00,0000,1111   ;3535           CID/READ.KMX,ID.ADDR/0E,J/<.>   ; Six bits are used.
                                ;3536
                                ;3537
                                ;3538
                                ;3539   ;S E C O N D A R Y    W C S    V E R S I O N
                                ;3540   ;The 4 lsb's of the ID address are the WCS sub-version(sixteen versions of
                                ;3541   ;WCS are possible for each FPLA version). The next two bits of the ID address
                                ;3542   ;indicate which PCS version is correct for this version of WCS. The console
                                ;3543   ;compares these two bits (ID.ADDR<5:4>) with the PCS version and reports a
                                ;3544   ;mismatch fatal error if they are different.
                                ;3545
                                ;3546   1112:    ;-------------------------------; Loop with version number as ID address
U 1112, 0000,003C,4180,2C00,0000,1112   ;3547           CID/READ.KMX,ID.ADDR/10,J/<.>   ; Six bits are used.
```

```
                                    ;3548   .TOC "  PCS microcode patches : WCS locations reserved for the micro-debugger"
                                    ;3549
                                    ;3550   ;This can go away when the .REGION pseudo-Op gets fixed
                                    ;3551   1120:  ;-------------------------------;
U 1120, 0000,003C,0180,F800,0000,1120 ;3552         J/<.>                          ;
                                    ;3553
                                    ;3554   1121:  ;-------------------------------;
U 1121, 0000,003C,0180,F800,0000,1121 ;3555         J/<.>                          ;
                                    ;3556
                                    ;3557   1122:  ;-------------------------------;
U 1122, 0000,003C,0180,F800,0000,1122 ;3558         J/<.>                          ;
                                    ;3559
                                    ;3560   1123:  ;-------------------------------;
U 1123, 0000,003C,0180,F800,0000,1123 ;3561         J/<.>                          ;
                                    ;3562
                                    ;3563   1124:  ;-------------------------------;
U 1124, 0000,003C,0180,F800,0000,1124 ;3564         J/<.>                          ;
                                    ;3565
                                    ;3566   1125:  ;-------------------------------;
U 1125, 0000,003C,0180,F800,0000,1125 ;3567         J/<.>                          ;
                                    ;3568
                                    ;3569   1126:  ;-------------------------------;
U 1126, 0000,003C,0180,F800,0000,1126 ;3570         J/<.>                          ;
                                    ;3571
                                    ;3572   1127:  ;-------------------------------;
U 1127, 0000,003C,0180,F800,0000,1127 ;3573         J/<.>                          ;
                                    ;3574
                                    ;3575   1128:  ;-------------------------------;
U 1128, 0000,003C,0180,F800,0000,1128 ;3576         J/<.>                          ;
                                    ;3577
                                    ;3578   1129:  ;-------------------------------;
U 1129, 0000,003C,0180,F800,0000,1129 ;3579         J/<.>                          ;
                                    ;3580
                                    ;3581   112A:  ;-------------------------------;
U 112A, 0000,003C,0180,F800,0000,112A ;3582         J/<.>                          ;
                                    ;3583
                                    ;3584   112B:  ;-------------------------------;
U 112B, 0000,003C,0180,F800,0000,112B ;3585         J/<.>                          ;
                                    ;3586
                                    ;3587   112C:  ;-------------------------------;
U 112C, 0000,003C,0180,F800,0000,112C ;3588         J/<.>                          ;
                                    ;3589
                                    ;3590   112D:  ;-------------------------------;
U 112D, 0000,003C,0180,F800,0000,112D ;3591         J/<.>                          ;
                                    ;3592
                                    ;3593   112E:  ;-------------------------------;
U 112E, 0000,003C,0180,F800,0000,112E ;3594         J/<.>                          ;
                                    ;3595
                                    ;3596   112F:  ;-------------------------------;
U 112F, 0000,003C,0180,F800,0000,112F ;3597         J/<.>                          ;
```

```
                                        ;3598   .TOC    "       PCS microcode patches : Patches prior to WCS116"
                                        ;3599
                                        ;3600   ;Date   : 21-JUL-77                                          Patch no. 001
                                        ;3601   ;Author : Paul R. Guilbault(04-APR-80), Unknown
                                        ;3602   ;
                                        ;3603   ;Source code file  : FLOAT
                                        ;3604   ;First included in :
                                        ;3605   ;
                                        ;3606   ;PROM address to be intercepted :  059D
                                        ;3607   ;
                                        ;3608   ;Description of problem :
                                        ;3609   ;21-JUL-77  IF THE LIMIT IN ACBF/D HAS ZERO EXPONENT BUT NON-ZERO FRACTION (I.E.
                                        ;3610   ;           A DIRTY 0), THE BRANCH TEST MAY NOT WORK. ALSO, IN COMPARING INDEX
                                        ;3611   ;           WITH LIMIT, THE EQUALITY CASE IS MISPLACED (SHOULD BE WITH CARRY=1).
                                        ;3612   ;04-APR-80  Add fix for ACBD branching on GTR when ADDEND is negative.
                                        ;3613   ;
                                        ;3614   ;Instead of that microinstruction, insert here the code which works :
                                        ;3615
;3616   .NOBIN
;3617   ;Algorithm to compare the INDEX to the LIMIT :
;3618   ;
;3619   ; 1.) Determine if the signs are different. If they are then : either   Index +, Limit -,       Index > Limit
;3620   ;                                                             or       Index -, Limit +,       Index < Limit
;3621   ;     Branch on the XOR of the SIGN(Addend) and SIGN(Index).
;3622   ;
;3623   ;                                    ADDEND   INDEX   XOR
;3624   ;     Addend is +, Branch on LEQ       +        +       0     Index is +, Index > Limit, no BRANCH
;3625   ;     Addend is +, Branch on LEQ       +        -       1     Index is -, Index < Limit,    BRANCH
;3626   ;     Addend is -, Branch on GEQ       -        +       1     Index is +, Index > Limit,    BRANCH
;3627   ;     Addend is -, Branch on GEQ       -        -       0     Index is -, Index < Limit, no BRANCH
;3628   ;
;3629   ;
;3630   ; 2.) If the signs are the same, then the exponent and fractions bits are compared in decreasing order of significance
;3631   ;     to determine the relationship between the magnitude of the INDEX and LIMIT.
;3632   ;
;3633   ;       a.) If the absolute INDEX = the absolute LIMIT, branch always.
;3634   ;
;3635   ;       b.) If the absolute INDEX > the absolute LIMIT, branch on the XOR of the SIGN(Addend) and SIGN(Index).
;3636   ;
;3637   ;       c.) If the absolute INDEX < the absolute LIMIT, branch on the EQV of the SIGN(Addend) and SIGN(Index).
;3638   ;
;3639   ; ABS(Index) .GTR. ABS(Limit) |                | SIGN(Addend)  SIGN(Index) | XOR |
;3640   ;      +6             +4      | INDEX.GTR.LIMIT |      +            +       |  0  | Addend is +, branch on LEQ, no BRANCH
;3641   ;      -6             -4      | INDEX.LSS.LIMIT |      +            -       |  1  | Addend is +, branch on LEQ,    BRANCH
;3642   ;      +6             +4      | INDEX.GTR.LIMIT |      -            +       |  1  | Addend is -, branch on GEQ,    BRANCH
;3643   ;      -6             -4      | INDEX.LSS.LIMIT |      -            -       |  0  | Addend is -, branch on GEQ, no BRANCH
;3644   ;
;3645   ;
;3646   ; ABS(Index) .LSS. ABS(Limit) |                | SIGN(Addend)  SIGN(Index) | EQV |
;3647   ;      +4             +6      | INDEX.LSS.LIMIT |      +            +       |  1  | Addend is +, branch on LEQ,    BRANCH
;3648   ;      -4             -6      | INDEX.GTR.LIMIT |      +            -       |  0  | Addend is +, branch on LEQ, no BRANCH
;3649   ;      +4             +6      | INDEX.LSS.LIMIT |      -            +       |  0  | Addend is -, branch on GEQ, no BRANCH
;3650   ;      -4             -6      | INDEX.GTR.LIMIT |      -            -       |  1  | Addend is -, branch on GEQ,    BRANCH
;3651   ;
;3652   ; *** Note : ALU/XOR sets ALU C31 like a subtract does but sets the remaining ALU conditions as would be expexted from an XO
```

```
                                ;3653   .BIN    ;                                       Continuation of Patch no. 001
                                ;3654   ;  The following code is the compare for ACBF and ACBD. Entry status :
                                ;3655   ;
                                ;3656   ;       D,Q     :       INDEX<H>        Index after add bits<31:00>
                                ;3657   ;
                                ;3658   ;       RC[T1]  :       INDEX<L>        Index after add bits<63:32>
                                ;3659   ;
                                ;3660   ;       RC[T3]  :       LIMIT<H>        Limit bits<31:00>
                                ;3661   ;
                                ;3662   ;       ID[T5]  :       LIMIT<H>        Limit bits<31:00>
                                ;3663   ;
                                ;3664   ;       ID[T6]  :       LIMIT<L>        Limit bits<63:32>
                                ;3665   ;
                                ;3666   ;       SS      :                       SIGN(Addend).XOR.SIGN(Index)
                                ;3667   ;
                                ;3668   ;       RC[T7]  :                       Branch address(PC+disp)
                                ;3669
                                ;3670   1140:   ;-------------------------------;
U 1140, 0010,0038,0180,F988,0000,1015   ;3671           RC[T1]_LC                       ;
                                ;3672
                                ;3673           ;-------------------------------;
                                ;3674           ALU_RC[T3],SC_ALU(EXP),         ; LOAD EXPONENT OF LIMIT IN SC
U 1015, 0010,0038,0580,F918,0197,7016   ;3675           FE_K[.1],CLK.UBCC               ; CLEAR EALU CC
                                ;3676
                                ;3677           ;-------------------------------;
                                ;3678           D_Q.XOR.RC[T3],                 ; GET DIFFERENCE BETWEEN INDEX AND LIMIT
                                ;3679           WORD,CLK.UBCC,                  ; TESTING ONLY SIGN, EXP, AND MSB'S
                                ;3680           SD_SS,                          ; SS&SD GET ADDEND<SIGN>.xor.INDEX<SIGN>
                                ;3681           FE_K[.1],CLK.UBCC,              ; CLEAR EALU CC
U 1016, 0811,6C20,0584,F918,0114,7053   ;3682           SC.NE.0?                        ; TEST EXPONENT OF LIMIT
                                ;3683
                                ;3684   =011    ;011----------------------------; SC.EQL.0, LIMIT IS ZERO
                                ;3685           ALU_0(A),RC[T3]_ALU,            ; CLEAN UP LIMIT<H>
U 1053, 0F03,003C,0180,F998,0000,1018   ;3686           D_0,J/ACBF.01                   ; NEED TO CLEAR LIMIT<L> AS WELL
                                ;3687
                                ;3688   ACBF.00.
                                ;3689           ;111----------------------------; SC.NEQ.0, LIMIT IS NOT 0
                                ;3690           SC_K[ZERO],                     ; CLEAR SC FOR LATER BRANCHING
                                ;3691           D_D.OXT[WORD].XOR.Q,            ; D GETS INDEX(H)<31:16>'LIMIT(H)<15:0>
U 1057, 081F,5B20,1980,F800,0084,7063   ;3692           ALU?,J/ACBF.02                  ; TEST DIFFERENCE <15:0>
                                ;3693   =
                                ;3694   ACBF.01:
                                ;3695           ;-------------------------------;
                                ;3696           ID[T6]_D,                       ; CLEAR LIMIT<L>
                                ;3697           D_Q.XOR.RC[T3],                 ; RE-COMPARE WITH CLEAN ZERO
                                ;3698           WORD,CLK.UBCC,                  ; TEST ONLY SIGN, EXP, AND MBS'S
                                ;3699           EALU_FE,                        ; CLEAR EALU CC
U 1018, 0811,6020,D980,3D18,0010,7057   ;3700           J/ACBF.00                       ;
```

```
                                        ;3701   =0011;                                          Continuation of Patch no. 001
                                        ;3702   ACBF.02:
                                        ;3703           ;0011-------------------------; EQUAL SIGNS, DIFFERENT BITS<14:0>
U 1063, 001D,2020,0180,F800,0010,1067   ;3704           ALU_Q.XOR.D,CLK.UBCC          ; |INDEX|.neq.|LIMIT|, RE-COMPARE
                                        ;3705                                         ; INDEX<15:0> TO LIMIT<15:0> TO SET C31
                                        ;3706
                                        ;3707           ;0111-------------------------; EQUAL SIGNS, EQUAL BITS<14:0>
                                        ;3708           ALU_Q.XOR.LC,LONG,CLK.UBCC,   ; COMPARE INDEX<H> TO LIMIT<H> <31:0>
                                        ;3709           EALU_FE,                      ; SD GETS SIGN(ADDEND).EQV.SIGN(INDEX)
                                        ;3710           SD_NOT.SD,                    ; MAY HAVE TO COMPARE LOW FRACTIONS
                                        ;3711           Q_ID[T6],                     ; COMPARE UPPER WORD IF LOWER IS EQUAL,
U 1067, 0011,2320,D9F3,2C00,0010,7050   ;3712           C31?,J/ACBF.04                ; OR LOWER WORD IF DIFFERENT
                                        ;3713   ACBF.03:
                                        ;3714           ;1011-------------------------; DIFFERENT SIGNS, DIFFERENT BITS<14:0>
                                        ;3715           CLR.IB.OPC,PC_PC+1,           ; SIGN(LIMIT).NE.SIGN(INDEX)
U 1068, C000,123C,0180,F804,4000,05A4   ;3716           EALU?,J/ACBF.7                ; BRANCH IFF SGN(ADDEND).NE.SGN(INDEX)
                                        ;3717   =
                                        ;3718   =0*
                                        ;3719   ACBF.04:
                                        ;3720           ;0*-------------------------; C31 = 0
U 1050, 0000,003C,0182,F800,0000,1068   ;3721           SS_SD,J/ACBF.03               ; |INDEX| .LSS. |LIMIT|
                                        ;3722
                                        ;3723           ;1*-------------------------; C31 = 1, |INDEX<H>| .GEQ. |LIMIT<H>|
                                        ;3724           D_Q.XOR.RC[T1],               ; COMPARE LIMIT<L> TO INDEX<L>
                                        ;3725           CLK.UBCC,WORD,                ; COMPARE ONLY BITS<47:32>
U 1052, 0811,6120,0180,F908,0010,704C   ;3726           EALU_FE,Z?                    ; KEEP EALU CC'S 0
                                        ;3727
                                        ;3728   =0      ;0-------------------------; Z = 0
                                        ;3729           CLR.IB.OPC,PC_PC+1,           ; |INDEX| .GTR. |LIMIT|
U 104C, C000,123C,0180,F804,4000,05A4   ;3730           EALU?,J/ACBF.7                ; ACBF.7
                                        ;3731
                                        ;3732           ;1-------------------------; Z = 1
                                        ;3733           SS_SD,                        ; |INDEX<H> .EQL. |LIMIT<H>|
                                        ;3734           D_D.OXT[WORD].XOR.Q,           ; D GETS LIMIT<63:48>'INDEX<47:32>
U 104D, 081F,5720,0182,F800,0000,1080   ;3735           STATE0?                       ; TEST FOR DOUBLE OR SINGLE
```

```
                                        ;3736  ;                                                Continuation of Patch no. 001
                                        ;3737  =0      ;0------------------------------; ACBD, ¦INDEX<H>¦ .EQL. ¦LIMIT<H>¦
                                        ;3738          ALU_Q.XOR.D,CLK.UBCC,LONG,      ; COMPARE LIMIT<47:32> WITH INDEX
                                        ;3739          EALU_FE,                        ; KEEP EALU CC'S CLEAR
                                        ;3740          SD_NOT.SD,                      ; SD GETS SIGN(ADDEND).XOR.SIGN(INDEX)
U 1080, 001D,3B20,0183,F800,0010,7 J0A  ;3741          ALU?,J/.ACBF.10                 ;
                                        ;3742
                                        ;3743          ;1------------------------------; ACBF, INDEX .EQL. LIMIT
U 1081, 2010,0038,0180,F939,4200,00AB   ;3744          PC&VA_RC[T7],FLUSH.IB,J/IB.FILL ; BRANCH
                                        ;3745  =1010
                                        ;3746  .ACBF.10:
                                        ;3747          ;1010----------------------------; Z=0(47:32 not=), C31=0(no meaning)
U 100A, 0000,033C,0180,F800,0000,1058   ;3748          C31?,J/.ACBF.8                  ; ¦LIMIT<47:32>¦ .NEQ. ¦INDEX<47:32>¦
                                        ;3749
                                        ;3750          ;1011----------------------------; Z=0(47:32 not=), C31=1(no meaning)
U 100B, 0000,033C,0180,F800,0000,1058   ;3751          C31?,J/.ACBF.8                  ; ¦LIMIT<47:32>¦ .NEQ. ¦INDEX<47:32>¦
                                        ;3752
                                        ;3753          ;1110----------------------------; Z=1(47:32 =), C31=0(valid for 63:32)
                                        ;3754          SS_SD,                          ; ¦LIMIT¦ .LSS. ¦INDEX¦
U 100E, 0000,003C,0182,F800,0000,1088   ;3755          J/ACBF.9                        ; SS GETS ADDEND<SIGN>.XOR.INDEX<SIGN>
                                        ;3756
                                        ;3757          ;1111----------------------------; Z=1(47:32 =), C31=1(valid for 63:32)
                                        ;3758          ALU_Q.XOR.RC[T1],               ; ¦LIMIT¦ .GEQ. ¦INDEX¦
                                        ;3759          CLK.UBCC,LONG,                  ; COMPARE LIMIT<L> to INDEX<L> ENTIRE
U 100F, 0011,2020,0180,F908,0010,7021   ;3760          EALU_FE                         ; LWORD TO SET Z, KEEP EALU CC CLEAN
                                        ;3761
                                        ;3762          ;--------------------------------;
U 1021, 0000,013C,0180,F800,0000,1088   ;3763          Z?                              ; IF 0 THEN EQL, IF NOT THEN GTR
                                        ;3764  =0
                                        ;3765  ACBF.9: ;0------------------------------; ¦LIMIT¦ .GTR. ¦INDEX¦
                                        ;3766          CLR.IB.OPC,PC_PC+1,             ; BRANCH ON SIGN(ADDEND).EQV.SIGN(INDEX)
U 1088, C000,123C,0180,F804,4000,05A4   ;3767          EALU?,J/ACBF.7                  ; ACBF.7
                                        ;3768
                                        ;3769          ;1------------------------------; LIMIT.EQL.INDEX - ALWAYS BRANCH
U 1089, 2010,0038,0180,F939,4200,00AB   ;3770          PC&VA_RC[T7],FLUSH.IB,J/IB.FILL ;
                                        ;3771
                                        ;3772  =0*
                                        ;3773  .ACBF.8:;0*-----------------------------; C=0, ¦LIMIT¦ .LSS. ¦INDEX¦
U 1058, 0000,003C,0182,F800,0000,105A   ;3774          SS_SD                           ; SS GETS SIGN(ADDEND).XOR.SIGN(INDEX)
                                        ;3775
                                        ;3776          ;1*------------------------------; C=1, ¦LIMIT¦ .GTR. ¦INDEX¦
                                        ;3777          CLR.IB.OPC,PC_PC+1,             ; (cannot be =)
U 105A, C000,123C,0180,F804,4000,05A4   ;3778          EALU?,J/ACBF.7                  ; ACBF.7
```

```
                                        ;3779   ;Date    : 21-JUL-77                                                    Patch no. 002
                                        ;3780   ;Author : T. FOSSUM
                                        ;3781   ;
                                        ;3782   ;Source code file  : DECMAL
                                        ;3783   ;First included in :
                                        ;3784   ;
                                        ;3785   ;PROM address to be intercepted : 01B4
                                        ;3786   ;
                                        ;3787   ;Description of problem :
                                        ;3788   ;       CVTLP-INSTRUCTION LEAVES DST-ADDRESS IN R1. RATHER THAN IN R3.
                                        ;3789   ;       ANY CVTLP-INSTRUCTION WITH NON-ZERO DST-ADDRESS WILL FAIL.
                                        ;3790   ;
                                        ;3791   ;Instead of that microinstruction, insert here the code which works :
                                        ;3792
                                        ;3793   1141:   ;-------------------------------;
U 1141, 0000,173C,0180,F800,0000,1054   ;3794           STATE3-0?                       ; TEST SIGN-BIT
                                        ;3795
                                        ;3796   =10*    ;10*----------------------------; BRANCH ON SIGN-BIT OF STATE
                                        ;3797           ALU_0(A),R[R1]_ALU,             ; CLEAR R1
U 1054, 0003,163C,0180,FA88,0000,0A93   ;3798           STATE7-4?,J/FINI3               ; TEST FOR OVERFLOW
                                        ;3799
                                        ;3800           ;11*----------------------------;
                                        ;3801           ALU_0(A),R[R1]_ALU,             ; CLEAR R1
U 1056, 0003,1A3C,0180,FA88,0000,089B   ;3802           PSL.CC?,J/FINI16                ; TEST PSL Z-BIT
```

```
                                ;3803   ;Date    : 02-AUG-77                                                    Patch no. 003
                                ;3804   ;Author : T. FOSSUM
                                ;3805   ;
                                ;3806   ;Source code file  : FLOAT
                                ;3807   ;First included in :
                                ;3808   ;
                                ;3809   ;PROM address to be intercepted : 06C3
                                ;3810   ;
                                ;3811   ;Description of problem :
                                ;3812   ;       COMMAS WERE LEFT OUT OF MICRO-INSTRUCTIONS USED IN
                                ;3813   ;       ACBD-INSTRUCTION. CAUSES INCORRECT RETURN-ADDRESSES.
                                ;3814   ;       ALSO, PC IS INCREMENTED BY ONE RATHER THAN BY TWO AS IT SHOULD.
                                ;3815   ;       ALSO, A SUBTRACION WAS DONE RATHER THAN AN ADDITION.
                                ;3816   ;       ALSO, NO CHECK IS MADE FOR RESERVED OPERAND ON LIMIT.
                                ;3817   ;       ALSO, WHEN THE ADDEND IS 0, ADDEND<L> IS SUBSTITUTED FOR INDEX<L>
                                ;3818   ;
                                ;3819   ;Instead of that microinstruction, insert here the code which works :
                                ;3820
                                ;3821   1142:   ;-------------------------------;
                                ;3822           RC[7]_Q+PC+1,                   ; CALCULATE BRANCH ADDRESS
                                ;3823           PC_PC+1,                        ; INCREMENT PC AGAIN
                                ;3824           CLR.IB.SPEC,                    ; CLEAR 1. BYTE OF B-DST
U 1142, D015,2010,D5F0,2DBC,0000,102D   ;3825           Q_ID[T5]                        ; GET LIMIT<H>
                                ;3826
                                ;3827           ;-------------------------------;
                                ;3828           ALU_Q,CHK.FLT.OPR,              ; CHECK LIMIT FOR RESERVED OP
                                ;3829           Q_D,                            ; COPY INDEX<L> FOR SWAP
U 102D, 0001,203C,65E0,F800,0884,70A8   ;3830           SC_K[.10]                       ; SETUP SC FOR SWAP OF HALVES
                                ;3831   =0**00
                                ;3832           ;0**00----------------------------;
                                ;3833           D_DAL.SC,                       ; SWAP HALVES OF INDEX
                                ;3834           SC_K[.FFF9],                    ; GET -7 FOR SHIFT
U 10A8, DD00,003D,BD80,F800,0084,66CA   ;3835           CLR.IB.SPEC,CALL,J/UNPACK       ; CLEAR 2. BYTE OF BDEST
                                ;3836
                                ;3837           ;0**01----------------------------; INDEX = 0
                                ;3838           Q_ID[T7],                       ; GET ADDRESS OF INDEX IF MEMORY
                                ;3839           LC_RC[T1],                      ; GET ADDEND<L> TO LATCH
U 10A9, 0000,163C,DDF0,2D08,0000,06D2   ;3840           STATE4?,J/ACBD.6                ; TEST FOR STORAGE PLACE
                                ;3841
                                ;3842           ;0**10----------------------------;
                                ;3843           Q_ID[T7],                       ; GET ADDRESS OF INDEX IF MEMORY
                                ;3844           LC_RC[T3],                      ; GET INDEX<L> TO LATCH
U 10AA, C000,163C,DDF0,2D18,0000,06D2   ;3845           STATE4?,J/ACBD.6                ; TEST FOR STORAGE PLACE
                                ;3846
                                ;3847           ;0**11----------------------------;
                                ;3848           ALU_LA.XOR.LC,SS_ALU15,         ;
                                ;3849           FE_NABS(SC-LA(EXP)),            ; CALCULATE SHIFT-AMOUNT
                                ;3850           CLR.UBCC,                       ; NOTE ITS DIRECTION
U 10AB, 0010,0021,0181,F800,0118,E583   ;3851           CALL,J/ADDD.6                   ; GO FINISH ADD
                                ;3852
                                ;3853   =1**11
                                ;3854           ;1**11----------------------------;
                                ;3855           Q_ID[T7],                       ; GET ADDRESS OF INDEX IF MEMORY
                                ;3856           LC_RC[T1],                      ; GET INDEX<L> TO LATCH
U 10BB, 0000,163C,DDF0,2D08,0000,06D2   ;3857           STATE4?,J/ACBD.6                ; TEST FOR STORAGE PLACE
```

```
                                    ;3858   ;Date    : 22-JUL-77                                                    Patch no. 004
                                    ;3859   ;Author : M. HURLEY
                                    ;3860   ;
                                    ;3861   ;Source code file  : CHAR
                                    ;3862   ;First included in :
                                    ;3863   ;
                                    ;3864   ;PROM address to be intercepted : 0803
                                    ;3865   ;
                                    ;3866   ;Description of problem :
                                    ;3867   ;       MATCHC HAS A STATE REGISTER COMBINATION THAT IS NOT SPECIFICALLY
                                    ;3868   ;       EXCLUDED ON FPD RESTART.
                                    ;3869   ;       THIS CAN ONLY BE INDUCED BY CHANGING BITS <25:24> OF R0 TO 10
                                    ;3870   ;       BETWEEN SAVING REGISTERS FOR FPD AND CONTINUING THE MATCHC
                                    ;3871   ;
                                    ;3872   ;Instead of that microinstruction, insert here the code which works :
                                    ;3873
                                    ;3874   1143:   ;-------------------------------;
                                    ;3875           VA_R[R1],                       ; PREPARE TO REREAD OBJ
U 1143, 0000,173C,0180,FA08,0200,1070 ;3876         BEN/STATE3-0                    ; BRANCH ACCORDING TO WHERE INTERRUPTED
                                    ;3877   =**00
                                    ;3878           ;**00-----------------------; <1:0> OF STATE
                                    ;3879           D[BYTE]_CACHE,                  ; RE-READ OUTER BYTE
                                    ;3880           STATE_OUTER,                    ; WAS IN OUTER LOOP
U 1070, 0000,803C,1980,4000,1404,6A6D ;3881         J/MATOUT1                       ; J/MATOUT1
                                    ;3882
                                    ;3883           ;**01-----------------------;
                                    ;3884           D[BYTE]_CACHE,                  ; RE-READ OBJ BYTE
                                    ;3885           STATE_INNEROBJ,                 ; WAS READING OBJ IN INNER LOOP
                                    ;3886           LAB_R[R3],                      ;
U 1071, 0000,803C,0580,4218,1404,6A75 ;3887         J/MATIN3                        ; J/MATIN3
                                    ;3888
                                    ;3889           ;**10-----------------------;
U 1072, 0000,003C,0180,F800,2000,0106 ;3890         J/RSVOPR,CLR.FPD                ; THEORETICALLY AN ILLEGAL CASE
                                    ;3891
                                    ;3892           ;**11-----------------------;
                                    ;3893           R[R3]_D-K[.1],                  ; DECREMENT SRC ADDR
U 1073, 001C,0000,0580,FA98,0000,06A4 ;3894         J/MATIN5                        ; J/MATIN5
```

```
                                        ;3895   ;Date    : 27-JUL-77                                                    Patch no. 005
                                        ;3896   ;Author : T. FOSSUM
                                        ;3897   ;
                                        ;3898   ;Source code file  : PROBE
                                        ;3899   ;First included in :
                                        ;3900   ;
                                        ;3901   ;PROM address to be intercepted : 0D10
                                        ;3902   ;
                                        ;3903   ;Description of problem :
                                        ;3904   ;       THE VA-MUX IS UNSTABLE BECAUSE OF IB-STALLS DURING A
                                        ;3905   ;       MICRO-BRANCH ON VA<31:30>.
                                        ;3906   ;       POPR-INSTRUCTION WITH CACHE TURNED OFF, MAY HAVE THIS PROBLEM.
                                        ;3907   ;
                                        ;3908   ;Instead of that microinstruction, insert here the code which works :
                                        ;3909
                                        ;3910   1144:    ;-------------------------------;
U 1144, 0000,003C,BD80,3C00,0000,10D0   ;3911           ID[Q.SV]_D                      ; SAVE Q-REGISTER
                                        ;3912
                                        ;3913   =00      ;00-------------------------------;
                                        ;3914           RC[SC.SV]_K[SC],                ; SAVE SC
                                        ;3915           MCT/MEM.NOP,                    ; IDEA OF ECO
U 10D0, 0018,1C39,1D80,09E8,0000,023C   ;3916           CALL,PSL.MODE?,J/GET.PTE        ; TEST VA<31:30>
                                        ;3917
                                        ;3918           ;01-------------------------------;
                                        ;3919           SC_RC[SC.SV],                   ; RESTORE SAVED SC
                                        ;3920           TEST.TB.RCHK,                   ; CHECK FOR ACCESS VIOLATION
U 10D1, 0010,0038,0180,0168,0082,0A51   ;3921           J/TBF.R2
                                        ;3922
                                        ;3923           ;10-------------------------------;
                                        ;3924           Q_D.AND.K[.10].RIGHT,           ; SET ACCESS VIOLATION IF APPROPRIATE
U 10D2, 0059,0034,65C0,F800,0000,0EA8   ;3925           J7M.FLT.B                       ; GO TAKE THE FAULT
                                        ;3926   =
```

```
                                    ;3927   ;Date    : 27-JUL-77                                                    Patch no. 006
                                    ;3928   ;Author : T. FOSSUM
                                    ;3929   ;
                                    ;3930   ;Source code file  : PROBE
                                    ;3931   ;First included in :
                                    ;3932   ;
                                    ;3933   ;PROM address to be intercepted : 0D12
                                    ;3934   ;
                                    ;3935   ;Description of problem :
                                    ;3936   ;       THE VA-MUX IS UNSTABLE BECAUSE OF IB-STALLS DURING A
                                    ;3937   ;       MICRO-BRANCH ON VA<31:30>.
                                    ;3938   ;       POPR-INSTRUCTION WITH CACHE TURNED OFF, MAY HAVE THIS PROBLEM.
                                    ;3939   ;
                                    ;3940   ;Instead of that microinstruction, insert here the code which works :
                                    ;3941
                                    ;3942   1145:   ;-------------------------------;
U 1145, 0000,003C,BD80,3C00,0000,1138 ;3943         ID[Q.SV]_D                      ; SAVE Q-REGISTER
                                    ;3944
                                    ;3945   =00     ;00------------------------------;
                                    ;3946         RC[SC.SV]_K[SC],                  ; SAVE SC
                                    ;3947         MCT/MEM.NOP,                      ; IDEA OF ECO
U 1138, 0018,1C39,1D80,09E8,0000,023C ;3948         CALL,PSL.MODE?,J/GET.PTE          ; TEST VA<31:30>
                                    ;3949
                                    ;3950         ;01------------------------------;
                                    ;3951         TEST.TB.WCHK,                     ; CHECK FOR ACCESS VIOLATION
U 1139, 0000,003D,0180,1000,0000,0D6E ;3952         CALL,J/TEST.PTE.W                 ; COULD SAVE AN INSTRUCTION HERE
                                    ;3953
                                    ;3954         ;10------------------------------;
                                    ;3955         D_D.OR.K[.4],                     ; FLAG MEM MGMT FAULT
U 113A, 0819,0030,1180,F800,0000,0EA3 ;3956         J/M.FLT.A                         ; GO TAKE THE FAULT
                                    ;3957
                                    ;3958         ;11------------------------------;
                                    ;3959         PC&VA_RC[PC.SV],                  ; RESTORE PC
                                    ;3960         Q_ID[D.SV],                       ; GET SAVED D-REG
U 113B, 0010,003A,B9F0,2D61,2A00,0020 ;3961         C[R.NEST.ERR,RETURN20             ; GO BACK TO PROBE-POINT
```

```
                                    ;3962   ;Date    : 12-JUL-77                                                 Patch no. 007
                                    ;3963   ;Author :
                                    ;3964   ;
                                    ;3965   ;Source code file  : IE
                                    ;3966   ;First included in :
                                    ;3967   ;
                                    ;3968   ;PROM address to be intercepted : 0D07
                                    ;3969   ;
                                    ;3970   ;Description of problem :
                                    ;3971   ;       RESERVED OPERAND FAULTS WITH FIRST PART DONE SET ATTEMPT
                                    ;3972   ;       TO RESTORE THE GENERAL REGISTERS FROM THE RLOG STACK.
                                    ;3973   ;       THIS PRODUCES RANDOM RESULTS.  A CASE THAT WOULD FAIL IS:
                                    ;3974   ;       POLYF   (R0)+,(R1)+,(R2)+
                                    ;3975   ;       WHERE A COEFFICIENT (OTHER THAN THE FIRST ONE) WAS 8000
                                    ;3976   ;
                                    ;3977   ;Instead of that microinstruction, insert here the code which works :
                                    ;3978
                                    ;3979   1146:   ;-------------------------------;
                                    ;3980           Q_PC, RC[PC.SV]_PC,             ; SAVE PC WHERE BAKUP.PC NEEDS IT
U 1146, 0014,0038,01C0,F9E0,0000,0908 ;3981         J7RSVOP1                        ; & GO CALL IT
```

```
                                ;3982   ;Date    : 27-JUL-77                                                    Patch no. 008
                                ;3983   ;Author : T. FOSSUM/R. LARY
                                ;3984   ;
                                ;3985   ;Source code file  : FLOAT
                                ;3986   ;First included in :
                                ;3987   ;
                                ;3988   ;PROM address to be intercepted : 0627
                                ;3989   ;
                                ;3990   ;Description of problem :
                                ;3991   ;       WHEN THE LOWER BITS OF THE FRACTIONS IN CMPD ARE
                                ;3992   ;       COMPARED, THE ALU CC'S ARE NOT CLOCKED TO REFLECT RESULT.
                                ;3993   ;       ALSO, ONE OF THE COMPARES TESTS N INSTEAD OF C.
                                ;3994   ;       CMPD    WHEN THE OPERANDS HAVE SAME EXPONENENT AND
                                ;3995   ;       HIGH ORDER FRACTION-BITS, BUT DIFFERENT LOW ORDER BITS.
                                ;3996   ;
                                ;3997   ;Instead of that microinstruction, insert here the code which works :
                                ;3998
                                ;3999   1148:   ;-------------------------------;
U 1148, 0019,3B34,C1C0,F8C0,0000,1083   ;4000           Q_Q.AND.K[.FFFF], ALU?          ; ISOLATE BITS <47:32> FOR COMPARE
                                ;4001   =0011
                                ;4002           ;0011-----------------------------; BRANCH ON ALU Z AND N-BITS
                                ;4003           ALU_RC[T0],SET.CC(INST),        ;
U 1083, C010,C038,0180,F904,4070,0062   ;4004           CLR.IB.OPC,PC_PC+1,J/IRD        ;
                                ;4005
                                ;4006           ;0111-----------------------------;
                                ;4007           ALU_D.OXT[WORD]-Q, CLK.UBCC,    ; COMPARE MIDDLE BITS OF FRACT
U 1087, 001F,4000,C9F0,2C00,0010,1032   ;4008           Q_ID[T2], J/.CHECKL             ; RESTORE ORIGINAL LOW FRACTION
                                ;4009
                                ;4010           ;1011-----------------------------;
                                ;4011           ALU_R[R15].XOR.K[.8000],        ;
                                ;4012           SET.CC(INST),                   ;
U 108B, C018,C020,4580,FA7C,4070,0062   ;4013           CLR.IB.OPC,PC_PC+1,J/IRD        ;
                                ;4014   =
                                ;4015   .CHECKL:
                                ;4016           ;-------------------------------; COMPARE LOW FRACTIONS
U 1032, 001D,3B00,0180,F800,0010,102A   ;4017           ALU_Q-D, LONG, CLK.UBCC, ALU?   ; BRANCH ON MIDDLE
                                ;4018   =1010
                                ;4019           ;1010-----------------------------;
                                ;4020           ALU_RC[T0],SET.CC(INST),        ;
                                ;4021           CLR.IB.OPC,PC_PC+1,             ; SRC<M> > DST<M>, CC_SRC
U 102A, C010,C038,0180,F904,4070,0062   ;4022           J/IRD                           ;
                                ;4023
                                ;4024           ;1011-----------------------------;
                                ;4025           ALU_R[R15].XOR.K[.8000],        ;
                                ;4026           SET.CC(INST),                   ; DST<M> > SRC<M>, CC_-DST
U 102B, C018,C020,4580,FA7C,4070,0062   ;4027           CLR.IB.OPC,PC_PC+1,J/IRD        ;
                                ;4028   =1111
                                ;4029           ;1111-----------------------------;
U 102F, 0000,1B3C,0180,F800,0000,065A   ;4030           ALU?, J/CHECKF                  ; SRC<M>=DST<M> - TEST SRC<L>-DST<L>
                                ;4031   =
```

```
                                    ;4032   ;Date    : 02-AUG-77                                                  Patch no. 009
                                    ;4033   ;Author : T. FOSSUM
                                    ;4034   ;
                                    ;4035   ;Source code file  : FLOAT
                                    ;4036   ;First included in :
                                    ;4037   ;
                                    ;4038   ;PROM address to be intercepted : 0586
                                    ;4039   ;
                                    ;4040   ;Description of problem :
                                    ;4041   ;       IN ACBF-INSTRUCTION, THE CONDITION CODES DO NOT
                                    ;4042   ;       GET CLOCKED WHEN EITHER INDEX OR
                                    ;4043   ;       ADDEND IS ZERO.
                                    ;4044   ;
                                    ;4045   ;Instead of that microinstruction, insert here the code which works :
                                    ;4046
                                    ;4047   1149:   ;--------------------------------;
                                    ;4048           ALU_D,                          ; NEW INDEX
                                    ;4049           SET.CC(INST),                   ; SET PSL CONDITION CODES
U 1149, 0001,C03C,0180,F800,0070,0693 ;4050         J/FL.PA.9                       ;
```

```
                                        ;4051   ;Date   : 03-AUG-77                                                  Patch no. 010
                                        ;4052   ;Author : J. LEONARD
                                        ;4053   ;
                                        ;4054   ;Source code file  : RPPR
                                        ;4055   ;First included in :
                                        ;4056   ;
                                        ;4057   ;PROM address to be intercepted : 008B
                                        ;4058   ;
                                        ;4059   ;MODIFIED  20-JUN-1979 TO FIX PAGE FAULT CLEARING T ./R.SHORT
                                        ;4060   ;
                                        ;4061   ;Description of problem :
                                        ;4062   ;       RET INSTRUCTION NEVER RESTORES R0 OR R1, BECAUSE
                                        ;4063   ;       OF INCORRECTLY MASKING THE REGISTER MASK.
                                        ;4064   ;               CALLG #0,FOO
                                        ;4065   ;       FOO:    ^M<R1,R0>
                                        ;4066   ;               MCOM    R0,R0
                                        ;4067   ;               RET
                                        ;4068   ;
                                        ;4069   ;Instead of that microinstruction, insert here the code which works :
                                        ;4070
                                        ;4071   114A:   ;-------------------------------;
                                        ;4072   .RET:   LAB_R[FP],      VA_LA,          ; LOAD VA FROM FRAME POINTER
                                        ;4073           Q_ID[PSL],                      ; GET PSL TO REPLACE PSW
U 114A, 0000,003C,3DF0,2E68,0284,703A   ;4074           SC_K[.EF]
                                        ;4075
                                        ;4076           ;-------------------------------;
                                        ;4077           VA_VA+4, Q_Q.ANDNOT.K[.FFFF],   ; BUMP VA TO PSW/MASK, CLR PSL<L>
U 103A, 0019,2024,C1C0,F803,0084,904A   ;4078           SC_SC+K[.FFFF]                  ; SC IS NOW X^EE
                                        ;4079
                                        ;4080           ;-------------------------------;
                                        ;4081           D[LONG]_CACHE,                  ; READ IN PSW/MASK
U 104A, 0083,0010,0380,4180,0000,1055   ;4082           RC[T0]_0+MASK+1.RIGHT2, SI/MUL- ; RC[T0] GETS FFFFF000
                                        ;4083
                                        ;4084           ;-------------------------------;
                                        ;4085           ALU_D.OXT[BYTE]+Q, D_ALU, Q_D,  ; FORM NEW PSL IN D, Q<31:16> GETS MASK
U 1055, 081F,9814,65E0,F803,0084,703D   ;4086           VA_VA+4, SC_K[.10], D.BYTES?    ; SC GETS SHIFT CT, TEST PSW<15:8>=0
                                        ;4087
                                        ;4088   =1101
                                        ;4089           ;1101---------------------------; SAVE NEW PSL, GET MASK IN D
                                        ;4090           RC[T1]_D, D_DAL.SC, VA_VA+4,    ; SET UP SS TO GET MASK<15>
U 103D, 0D01,0D3C,0187,F98B,0000,1022   ;4091           SS_0&SD_0, Q31?, J/.RET.0       ; TEST MASK<15>
                                        ;4092
                                        ;4093           ;1111---------------------------;
U 103F, 0000,003C,0180,F800,0000,0106   ;4094           J/RSVOPR                        ; NASTY MAN - PAPA SPANK!
                                        ;4095
                                        ;4096   =01*                                    ; PSL<31>=0 (NOT COMPATIBILITY MODE)
                                        ;4097   .RET.0: ;01*--------------------------; MASK<15>=0
                                        ;4098           D_D.ANDNOT.RC[T0],STATE_D(EXP), ; D=MASK<11:0>, STATE=MASK<14:7>,
                                        ;4099           SC_D(EXP)(A),                   ; SC = MASK<14:7> ALSO
U 1022, 0811,0024,0180,F903,1498,6CC3   ;4100           VA_VA+4, CLK.UBCC, J/RET.1      ; SET Z ON MASK<11:0>
                                        ;4101
                                        ;4102           ;11*--------------------------; MASK<15> = 1
                                        ;4103           D_D.ANDNOT.RC[T0],STATE_D(EXP), ; D=MASK<11:0>, STATE=MASK<14:7>,
                                        ;4104           SD_NOT.SD, SC_D(EXP)(A),        ; SET SD TO MASK<15>, SC = MASK<14:7>
U 1026, 0811,0024,0183,F903,1498,6CC3   ;4105           VA_VA+4, CLK.UBCC, J/RET.1      ; SET Z ON MASK<11:0>
```

```
                                        ;4106   ;Date    : 10-AUG-77                                                    Patch no. 011
                                        ;4107   ;Author : R. LARY
                                        ;4108   ;
                                        ;4109   ;Source code file  : ARITH
                                        ;4110   ;First included in :
                                        ;4111   ;
                                        ;4112   ;PROM address to be intercepted : 0413
                                        ;4113   ;
                                        ;4114   ;Description of problem :
                                        ;4115   ;       INCORRECT ADDRESS CONSTRAINT IN EDIV CAUSES ACCESS VIOLATIONS
                                        ;4116   ;       WHEN QUOTIENT IS NEGATIVE.  ALSO, QUOTIENT OF -2**32 TREATED AS
                                        ;4117   ;       OVERFLOW.
                                        ;4118   ;       EXAMPLE(OF BOTH): EDIV #1,#FFFFFFFF80000000,R0,R1
                                        ;4119   ;
                                        ;4120   ;Instead of that microinstruction, insert here the code which works :
                                        ;4121
                                        ;4122   114B:   ;-------------------------------;
U 114B, 0019,0100,0580,F800,0050,1090   ;4123           ALU_D-K[.1], N&Z_ALU.V&C_0, Z?  ; SET UP OVFLO TEST, CHECK IF 0
                                        ;4124   =0
                                        ;4125           ;0-------------------------------; QUOTIENT < 0
                                        ;4126           D_0-D,N&Z_ALU.V&C_0,            ; NEGATE QUOTIENT
U 1090, 081F,3A00,0180,F800,0050,00E6   ;4127           PSL.N?,J/EDIV.9                 ; CHECK FOR OVFLO
                                        ;4128
                                        ;4129           ;1-------------------------------; QUOTIENT = 0
                                        ;4130           ID[T0]_D,D_Q,SC_0(A),STATE_0(A),; QUO = D, REM = Q
                                        ;4131           ALU_0(A), N&Z_ALU.V&C_0,        ; SET COND CODES FOR ZERO RESULT
U 1091, 0C03,003C,C180,3C00,54D8,708C   ;4132           INTRPT.STROBE, J/EMODF.11       ; USE EMODF CODE TO STORE RESULTS
                                        ;4133   =
```

```
                                        ;4134   ;Date    : 10-AUG-77                                                    Patch no. 02
                                        ;4135   ;Author : R. LARY
                                        ;4136   ;
                                        ;4137   ;Source code file  : FLOAT
                                        ;4138   ;First included in :
                                        ;4139   ;
                                        ;4140   ;PROM address to be intercepted : 04F4
                                        ;4141   ;
                                        ;4142   ;Description of problem .
                                        ;4143   ;       IF CVTDF HAS TO ROUND A FLOATING POINT NUMBER WHOSE HIGH FRACTION
                                        ;4144   ;       IS ALL ONES, IT PRODUCES A RESULT WHICH IS ONE-HALF OF THE TRUE RESULT
                                        ;4145   ;       EXAMPLE: CVTDF #FFFF407FFFFFFFFF,R0 PRODUCES 00004000 IN R0
                                        ;4146   ;
                                        ;4147   ;Instead of that microinstruction, insert here the code which works :
                                        ;4148
                                        ;4149   114C:   ;-------------------------------;
                                        ;4150           FE_SC,D_D+K[.80],N&Z_ALU.V&C_0, ; SAVE EXP, ROUND FRACT
U 114C, 0819,1414,4180,F800,0150,1219   ;4151           SC?                             ; CK FOR 0
                                        ;4152
                                        ;4153   =*01    ;*01----------------------------; SOURCE WAS 0
                                        ;4154           D_0, ALU_K[ZERO], SET.CC(INST), ; THEREFORE DEST GETS 0
U 1219, FF18,C03B,19F0,F847,0070,0300   ;4155           WRITE.DEST                      ;
                                        ;4156
                                        ;4157           ;*11----------------------------; SOURCE NON-ZERO
U 121B, 0000,1A3C,0180,F800,0080,C434   ;4158           SC_SC+1, PSL.N?, J/CVTDF.2      ; BUMP EXP IN CASE OF OVFLO, TEST OVFLO
                                        ;4159   =
```

```
                                        ;4160   ;Date    : 11-AUG-77                                                    Patch no. 013
                                        ;4161   ;Author : T. FOSSUM
                                        ;4162   ;
                                        ;4163   ;Source code file  : DECMAI.
                                        ;4164   ;First included in :
                                        ;4165   ;
                                        ;4166   ;PROM address to be intercepted : 0917
                                        ;4167   ;
                                        ;4168   ;Description of problem :
                                        ;4169   ;       CVTPL-INSTRUCTION MAY NOT DETECT OVERFLOW IF
                                        ;4170   ;       SRC-STRING HAS LENGTH 11 AND LEADING DIGIT IS 8 OR 9.
                                        ;4171   ;
                                        ;4172   ;Instead of that microinstruction, insert here the code which works :
                                        ;4173
                                        ;4174   114D:   ;-------------------------------;
U 114D, 0803,803C,0180,F800,0000,1065   ;4175           ALU_D.OXT[BYTE],D_ALU           ;
                                        ;4176
                                        ;4177           ;-------------------------------;
U 1065, 0019,0000,0D80,F80C,0010,106A   ;4178           ALU_D-K[.3],CLK.UBCC            ; CLOCK WHOLE LONGWORD
                                        ;4179
                                        ;4180           ;-------------------------------;
                                        ;4181           ALU_D.OXT[BYTE].AND.K[.FFF0],   ; ISOLATE HIGH NIBBLE
                                        ;4182           Q_ALU.RIGHT2,                   ; SHIFTED RIGHT TWICE
U 106A, 0C9B,9B34,6DC0,F800,0000,0175   ;4183           ALU.N?,J/P2L3                   ; TEST COMPARISON
```

```
                                    ;4184   ;Date    : 15-AUG-77                                                    Patch no. 014
                                    ;4185   ;Author : T. FOSSUM
                                    ;4186   ;
                                    ;4187   ;Source code file  : DECMAL
                                    ;4188   ;First included in :
                                    ;4189   ;
                                    ;4190   ;PROM address to be intercepted : 0886
                                    ;4191   ;
                                    ;4192   ;Description of problem :
                                    ;4193   ;       IN THE CVTPL-INSTRUCTION, THE SRC-STRING -0
                                    ;4194   ;       WILL CAUSE THE V-BIT OF PSL TO BE SET.
                                    ;4195   ;       ALSO, IF THE DST IS A REGISTER, THE
                                    ;4196   ;       INCORRECT REGISTER # IS READ FROM RLOG.
                                    ;4197   ;
                                    ;4198   ;Instead of that microinstruction, insert here the code which works :
                                    ;4199
                                    ;4200   114E:   ;-------------------------------;
                                    ;4201           EALU_K[.1],CLK.UBCC,            ; CLEAR EALU CC
                                    ;4202           VA_Q,QK/RIGHT,                  ; LOAD DST-ADDRESS
U 114E, 0001,2D3C,0580,F800,0214,7005 ;4203         D.NE.0?                         ;
                                    ;4204
                                    ;4205   =101    ;101----------------------------; BRANCH ON D NE 0
                                    ;4206           SC_Q(EXP),                      ; LOAD REGISTER #
U 1005, 0001,203C,01F8,F800,0088,6B8B ;4207         Q_0,J/P2LF3                     ;
                                    ;4208
                                    ;4209           ;111----------------------------;
                                    ;4210           SC_Q(EXP),                      ; LOAD REGISTER #
U 1007, 0001,373C,01F8,F800,0088,66F8 ;4211         Q_0,STATE3-0?,J/P2LF2           ; TEST SIGN-BIT OF STATE
```

```
                                    ;4212   ;Date    : 28-SEP-77                                                    Patch no. 015
                                    ;4213   ;Author : R. LARY
                                    ;4214   ;
                                    ;4215   ;Source code file  : FLOAT
                                    ;4216   ;First included in :
                                    ;4217   ;
                                    ;4218   ;PROM address to be intercepted : 0253, 0255, and 0473
                                    ;4219   ;
                                    ;4220   ;Description of problem :
                                    ;4221   ;       INTEGER OVERFLOW RESULTING FROM CVTFB, CVTFW, CVTFL, CVTRFL,
                                    ;4222   ;       CVTDB, CVTDW, CVTDL, OR CVTRDL INSTRUCTIONS DOES NOT CAUSE IV TRAP.
                                    ;4223   ;       THE V BIT IS SET CORRECTLY BUT NO TRAP OCCURS.
                                    ;4224   ;
                                    ;4225   ;Instead of that microinstruction, insert here the code which works :
                                    ;4226
                                    ;4227   1152:   ;-------------------------------;
                                    ;4228           ALU_D,N&Z_ALU.V&C_0,DT/INST.DEP,; SET CONDITION CODES ON RESULT
U 1152, 0001,DA3C,0180,F800,0050,10AC ;4229         PSL.V?                          ; TEST FOR OVERFLOW
                                    ;4230   =1100
                                    ;4231   GOUT.1: ;1100---------------------------;
U 10AC, F000,003F,01F0,F847,0000,030U ;4232         WRITE.DEST                      ; NO OVERFLOW, GO STORE DEST
                                    ;4233   =1110
                                    ;4234   CVTOFL: ;1110---------------------------;
U 10AE, 0000,003D,31F0,2C00,0000,0DFC ;4235         Q_ID[CES], CALL[INOVFL]         ; SIGNAL INTEGER OVERFLOW
                                    ;4236   =1111
                                    ;4237           ;1111---------------------------;
U 10AF, F000,003F,01F0,F847,0000,0300 ;4238         WRITE.DEST                      ; WRITE THE RESULT ANYWAY
```

```
                                ;4239   ;Date    : 28-SEP-77                                                    Patch no. 016
                                ;4240   ;Author : R. LARY
                                ;4241   ;
                                ;4242   ;Source code file  : FORKS
                                ;4243   ;First included in :
                                ;4244   ;
                                ;4245   ;PROM address to be intercepted : 0181
                                ;4246   ;
                                ;4247   ;Description of problem :
                                ;4248   ;       THE ASHQ INSTRUCTION DOES NOT WORK CORRECTLY ON OVERFLOW.
                                ;4249   ;       THE V BIT IS SET CORRECTLY BUT NO TRAP OCCURS IF ENABLED.
                                ;4250   ;       **NOTE** THAT THIS FIX INTERCEPTS 2 PROM LOCATIONS(SEE NEXT PAGE)
                                ;4251   ;
                                ;4252   ;Instead of that microinstruction, insert here the code which works :
                                ;4253
                                ;4254   1153:   ;--------------------------------;
U 1153, 0000,003C,0180,F800,0020,10AE   ;4255           SET.V, J/CVTOFL                 ; OVERFLOW PREVIOUSLY DETECTED
```

```
                                    ;4256   ;Date    : 28-SEP-77                                                    Patch no. 017
                                    ;4257   ;Author : R. LARY
                                    ;4258   ;
                                    ;4259   ;Source code file  : FORKS
                                    ;4260   ;First included in :
                                    ;4261   ;
                                    ;4262   ;PROM address to be intercepted : 01A1
                                    ;4263   ;
                                    ;4264   ;Description of problem :
                                    ;4265   ;       THE ASHQ INSTRUCTION DOES NOT WORK CORRECTLY ON OVERFLOW.
                                    ;4266   ;       THE V BIT IS SET CORRECTLY BUT NO TRAP OCCURS IF ENABLED.
                                    ;4267   ;       **NOTE** THAT THIS FIX INTERCEPTS 2 PROM LOCATIONS(SEE PREV PAGE)
                                    ;4268   ;
                                    ;4269   ;Instead of that microinstruction, insert here the code which works :
                                    ;4270
                                    ;4271   1154:   ;--------------------------------;
U 1154, 0C00,003C,0180,F800,0020,10AE ;4272             D_Q, SET.V, J/CVTOFL            ; OVERFLOW PREVIOUSLY DETECTED
```

```
                                        ;4273   ;Date    : 06-OCT-77                                                    Patch no. 018
                                        ;4274   ;Author : T. FOSSUM
                                        ;4275   ;
                                        ;4276   ;Source code file  : IE
                                        ;4277   ;First included in :
                                        ;4278   ;
                                        ;4279   ;PROM address to be intercepted : 0E03
                                        ;4280   ;
                                        ;4281   ;Description of problem :
                                        ;4282   ;       ACBF AND ACBD CLEAR PSL C-BIT WHEN UNDERFLOW OR OVERFLOW OCCURS.
                                        ;4283   ;       NOTE 2 WORDS
                                        ;4284   ;
                                        ;4285   ;Instead of that microinstruction, insert here the code which works :
                                        ;4286
                                        ;4287   1155:   ;-- ----------------------------;
                                        ;4288           Q_ID[CES],                      ;  GET CPU ERROR/STATUS REG.
                                        ;4289           ALU_-1,N&Z_ALU,                 ;  CHANGED FROM CCL5, LOADS N AND Z
U 1155, 0003,0028,31F0,2C00,0060,0E08   ;4290           J/IE.PA.18                      ;
```

```
                                        ;4291   ;Date    : 06-OCT-77                                                    Patch no. 019
                                        ;4292   ;Author : T. FOSSUM
                                        ;4293   ;
                                        ;4294   ;Source code file  : IE
                                        ;4295   ;First included in :
                                        ;4296   ;
                                        ;4297   ;PROM address to be intercepted : 0E09
                                        ;4298   ;
                                        ;4299   ;Description of problem :
                                        ;4300   ;       CONTINUATION OF ACBF AND ACBD CLEAR PSL C-BIT WHEN UNDERFLOW OR
                                        ;4301   ;       OVERFLOW OCCURS.
                                        ;4302   ;
                                        ;4303   ;Instead of that microinstruction, insert here the code which works :
                                        ;4304
                                        ;4305   1156:   ;-------------------------------;
                                        ;4306           Q_ID[CES],                      ;  GET CPU ERROR/STATUS REG.
                                        ;4307           ALU_0(A),N&Z_ALU,               ;  CHANGED FROM CCL5, LOADS N AND Z
U 1156, 0003,003C,31F0,2C00,0060,0E0A   ;4308           J/IE.PA.18A                     ;
```

```
                                        ;4309   ;Date   : 20-SEP-77                                                  Patch no. 020
                                        ;4310   ;Author : M. HURLEY
                                        ;4311   ;
                                        ;4312   ;Source code file  : EDIT
                                        ;4313   ;First included in :
                                        ;4314   ;
                                        ;4315   ;PROM address to be intercepted : 0783
                                        ;4316   ;
                                        ;4317   ;Description of problem :
                                        ;4318   ;       EDITPC: IF SRC LENGTH = 0, THE PATTERN 'FILL'' DOES NO FILLS
                                        ;4319   ;       PATTADDR: ^X00000081(FILL 1 CHAR,EOSEND)
                                        ;4320   ;       EDITPC #0,SRCADDR,PATTADDR,DESTADDR
                                        ;4321   ;
                                        ;4322   ;Instead of that microinstruction, insert here the code which works :
                                        ;4323
                                        ;4324   1157:   ;--------------------------------; PATTERN = 81-8F
                                        ;4325           STATE_STATE.OR.FILL,            ; STATE<2:0>_7
                                        ;4326           ALU_K[.7],CLK.UBCC,DT/BYTE,     ; FORCE ALUCC<Z>=0 FOR Z TEST IN
U 1157, 0018,8038,5D80,FA28,1414,2AE4   ;4327           LAB_R[R5],J/EDFILL              ; FILL LOOP. LATCH UP LAST DEST ADDR
```

```
                                    ;4328   ;Date    : 07-OCT-77                                                Patch no. 021
                                    ;4329   ;Author : R. LARY
                                    ;4330   ;
                                    ;4331   ;Source code file  : FLOAT
                                    ;4332   ;First included in :
                                    ;4333   ;
                                    ;4334   ;PROM address to be intercepted : 061B
                                    ;4335   ;
                                    ;4336   ;Description of problem :
                                    ;4337   ;       WARM DIVF DOES NOT CLEAR Q BEFORE ENTERING DIVIDE LOOP
                                    ;4338   ;       THIS CAN CAUSE SPURIOUS ROUNDING IF EXCESS QUOTIENT BITS ARE
                                    ;4339   ;       BETWEEN 0111111 AND 1000000
                                    ;4340   ;
                                    ;4341   ;       DIVF #CFA4058,#A9CD3D3D SHOULD YIELD BBBE3D60, NOT BBBF3D60
                                    ;4342   ;
                                    ;4343   ;Instead of that microinstruction, insert here the code which works :
                                    ;4344
                                    ;4345   1158:   ;-------------------------------;
U 1158, 0000,003C,01F8,F800,0100,A3D8   ;4346           FE_SC-FE,Q_0,J/DIVF4            ; ECO TO CLEAR Q
```

```
                                ;4347   ;Date    : 07-OCT-77                                                    Patch no. 022
                                ;4348   ;Author : R. LARY
                                ;4349   ;
                                ;4350   ;Source code file  : FLOAT
                                ;4351   ;First included in :
                                ;4352   ;
                                ;4353   ;PROM address to be intercepted : 058B
                                ;4354   ;
                                ;4355   ;Description of problem :
                                ;4356   ;
                                ;4357   ;*****************************************************************
                                ;4358   ;* This patch is bypassed by and incoporated into patch 087. *
                                ;4359   ;*****************************************************************
                                ;4360   ;
                                ;4361   ;       WARM DIVF *-R DOES NOT CLEAR Q BEFORE ENTERING DIVIDE LOOP
                                ;4362   ;       THIS CAN CAUSE SPURIOUS ROUNDING IF EXCESS QUOTIENT BITS ARE
                                ;4363   ;       BETWEEN 0111111 AND 1000000
                                ;4364   ;
                                ;4365   ;       DIVF #CFA4058,R0 WHEN R0=A9CD3D3D SHOULD YIELD BBBE3D60, NOT BBBF3D60
                                ;4366   ;
                                ;4367   ;Instead of that microinstruction, insert here the code which works :
```

```
                                ;4368   ;Date    : 20-OCT-77                                                    Patch no. 023
                                ;4369   ;Author : CHUCK MATHIS
                                ;4370   ;
                                ;4371   ;Source code file  : ERCODE
                                ;4372   ;First included in :
                                ;4373   ;
                                ;4374   ;PROM address to be intercepted : 0F19
                                ;4375   ;
                                ;4376   ;Description of problem :
                                ;4377   ;       ERROR HANDLING CODE ROUTINE, PUSH PARAMETERS ON STACK & CHECK
                                ;4378   ;       FOR ERRORS, WILL HANG-UP ON NOT BUSY SOMETIMES BECAUSE SBI BUSY
                                ;4379   ;       IS BEING CHECKED ASSUMING WRONG POLARITY.
                                ;4380   ;        KEN OKIN'S S/W FINDS THE PROBLEM
                                ;4381   ;
                                ;4382   ;Instead of that microinstruction, insert here the code which works :
                                ;4383
                                ;4384   115A:   ;-------------------------------; PATTERN = 81-8F
                                ;4385           Q_ID[SBI.ERR],                  ; GET SBI ERROR REGISTER
U 115A, 0000,003C,65F0,2C00,0000,1095   ;4386   J/EH.PUSH8                      ;
                                ;4387   =0
                                ;4388   .EH.PUSH7:
                                ;4389           ;0-------------------------------; BRANCH ON ALU.Z (BEN/Z)
                                ;4390           SC_SC-K[.1],                    ; GO TO NEXT REGISTER
U 1094, 0000,003C,0580,F800,0084,AF10   ;4391   J/EH.PUSH                       ; GO & SEE IF SBI ERROR OCCURRED
                                ;4392
                                ;4393   EH.PUSH8
                                ;4394           ;1-------------------------------;***SBI  BUSY***
                                ;4395           ALU_Q.AND.K[.2],                ;
                                ;4396           CLK.UBCC,                       ; TEST TO SEE IF SBI BUSY
U 1095, 0019,2034,0980,F800,0010,1075   ;4397   J/.EH.PUSH9                     ;
                                ;4398
                                ;4399   .EH.PUSH9:
                                ;4400           ;-------------------------------;
                                ;4401           Q_ID[SBI.ERR],                  ; GET SBI ERROR REG AGAIN
U 1075, 0000,013C,65F0,2C00,0000,1094   ;4402   Z?,J/.EH.PUSH7                  ; WAS SBI NOT BUSY?
```

```
                                        ;4403   ;Date    : 22-NOV-77                                                    Patch no. 024
                                        ;4404   ;Author : T. FOSSUM
                                        ;4405   ;
                                        ;4406   ;Source code file  : DECMAL
                                        ;4407   ;First included in :
                                        ;4408   ;
                                        ;4409   ;PROM address to be intercepted : 0C82
                                        ;4410   ;
                                        ;4411   ;Description of problem :
                                        ;4412   ;       INTERRUPT BIT OF STATE-REGISTER CLOBBERS LOW BIT
                                        ;4413   ;       OF D-REGISTER IN SAVER0-ROUTINE FOR PACKED BCD INSTRUCTIONS.
                                        ;4414   ;
                                        ;4415   ;HOW CAN THIS ECO BE TESTED?
                                        ;4416   ;       RUN DIAGNOSTICS WHICH CAUSE INTERRUPTS INSIDE DIVP.
                                        ;4417   ;
                                        ;4418   ;Instead of that microinstruction, insert here the code which works :
                                        ;4419
                                        ;4420   115B:   ;-------------------------------;
U 115B, 0819,0024,4580,F800,0000,1085   ;4421           ALU_D.ANDNOT.K[.8000],D_ALU     ; MASK OUT INTERRUPT BIT
                                        ;4422
                                        ;4423           ;-------------------------------;
                                        ;4424           EALU_FE,                        ;
                                        ;4425           D_PACK.FP.LEFT,                 ;
U 1085, 0828,0038,0180,F800,0000,6C84   ;4426           J7DC.PA.24                      ;
```

```
                                ;4427   ;Date    : 23-NOV-77                                                    Patch no. 025
                                ;4428   ;Author : R. J. Avarbock(30-JUL-80), T. FOSSUM
                                ;4429   ;
                                ;4430   ;Source code file  : DECMAL
                                ;4431   ;First included in :
                                ;4432   ;
                                ;4433   ;PROM address to be intercepted : 096D
                                ;4434   ;
                                ;4435   ;Description of problem :
                                ;4436   ;       ADDP4 AND SUBP4 TESTS FOR INTERRUPTS AFTER CHANGING 2. OPERAND
                                ;4437   ;       THUS NOT RESTARTING PROPERLY.
                                ;4438   ;       Modify to fix v bit problem with negative numbers.
                                ;4439   ;
                                ;4440   ;HOW CAN THIS ECO BE TESTED?
                                ;4441   ;       RUN ADDP4 AND SUBP4 DIAGNOSTICS WITH INTERRUPTS.
                                ;4442   ;
                                ;4443   ;Instead of that microinstruction, insert here the code which works :
                                ;4444
                                ;4445   115C:   ;-------------------------------;
U 115C, 081D,0000,0180,F800,4000,108A   ;4446           ALU_D-Q,D_ALU,INTRPT.STROBE     ; DECIMAL ADJUST, STROBE INTERRUPTS
                                ;4447
                                ;4448           ;-------------------------------;
                                ;4449           RC[T7]_ALU,ALU_0(A),            ; INITIALIZE R7 FOR OVERFLOW
U 108A, 0003,003C,1980,F9B8,0134,70A0   ;4450           FE_K[ZERO]                      ; In case no mask gen'ed.   **MOD**
                                ;4451
                                ;4452           ;-------------------------------;
                                ;4453           STATE_STATE.ANDNOT.K[.10],      ; CLEAR NEGATE BIT
                                ;4454           ALU_LB,Q_ALU,                   ; GET DST-LENGTH
                                ;4455           BYTE,CLK.UBCC,                  ; CLOCK IT
U 10A0, 000C,8E38,65C0,F800,1414,504E   ;4456           BEN/INTERRUPT                   ; TEST FOR INTERRUPTS
                                ;4457
                                ;4458   =100*110
                                ;4459           ;100*110--------------------------;
                                ;4460           ALU_Q+K[.8],                    ; INCREMENT NIBBLE COUNT
                                ;4461           SHF7ALU.DT,LONG,                ; SHIFT IT LEFT TWICE
                                ;4462           SC_ALU,QK/SHF,                  ; LOAD IT IN Q AND SC
                                ;4463           CLK.UBCC,                       ; CLOCK NEW CONDITION CODES
                                ;4464           LC_RC[T7],SIGNS?,               ; GET OLD OVERFLOW
U 104E, 0079,2D15,01C0,F938,0092,0E59   ;4465           CALL,J/WRITE1                   ; JUMP TO WRITE1
                                ;4466
                                ;4467   =100*111
                                ;4468           ;100*111--------------------------;
                                ;4469           STATE_STATE.OR.K[.80],          ; SET INTERRUPT BIT
U 104F, 0000,003C,4180,F800,1404,201?   ;4470           J/ADS.MEMORY.FAULT              ; JUMP TO "ADS.MEMORY.FAULT"
                                ;4471
                                ;4472   =110*110
                                ;4473           ;110*110--------------------------;
                                ;4474           SC_K[ZERO],LAB_R[R0],Q_LB,      ; GET FIRST LENGTH
U 106E, 0F0C,0039,19C0,FA00,0084,6C26   ;4475           D_0,CALL,J/UPDATE               ; CALL UPDATE ROUTINE
                                ;4476
                                ;4477   =111*110
                                ;4478           ;111*110--------------------------;
                                ;4479           ALU_LA-K[.4],SC_SC+1,           ;
U 107E, 0018,0000,1180,F8E8,0080,C446   ;4480           R(SC)_ALU,J/DC.PA.25            ;
                                ;4481   =
```

```
                                    ;4482   ;Date    : 23-NOV-77                                                Patch no. 026
                                    ;4483   ;Author : T. FOSSUM
                                    ;4484   ;
                                    ;4485   ;Source code file  : DECMAL
                                    ;4486   ;First included in :
                                    ;4487   ;
                                    ;4488   ;PROM address to be intercepted : 0CE7
                                    ;4489   ;
                                    ;4490   ;Description of problem :
                                    ;4491   ;       MULP DOES NOT RETURN TO CORRECT SPOT AFTER AN INTERRUPT.
                                    ;4492   ;
                                    ;4493   ;HOW CAN THIS ECO BE TESTED?
                                    ;4494   ;       RUN MULP DIAGNOSTICS WITH INTERRUPTS.
                                    ;4495   ;
                                    ;4496   ;Instead of that micrcinstruction, insert here the code which works :
                                    ;4497
                                    ;4498   115D:   ;-------------------------------;
U 115D, 0000,003C,4180,F800,1404,30AD ;4499         STATE_STATE.OR.K[.80]           ;
                                    ;4500
                                    ;4501         ;-------------------------------;
                                    ;4502         STATE_STATE.OR.K[.1],           ;
U 10AD, 0000,003C,0580,F800,1404,2991 ;4503         J/MULT.MEMORY.FAULT             ;
```

```
                                        ;4504   ;Date    : 23-NOV-77                                                    Patch no. 027
                                        ;4505   ;Author : T. FOSSUM
                                        ;4506   ;
                                        ;4507   ;Source code file  : DECMAL
                                        ;4508   ;First included in :
                                        ;4509   ;
                                        ;4510   ;PROM address to be intercepted : 0B6D
                                        ;4511   ;
                                        ;4512   ;Description of problem :
                                        ;4513   ;       CVTPT AND CVTPS DO NOT CLEAR OVERFLOW ON RESTART.
                                        ;4514   ;
                                        ;4515   ;HOW CAN THIS ECO BE TESTED?
                                        ;4516   ;       RUN CVTPT DIAGNOSTICS WITH INTERRUPTS.
                                        ;4517   ;
                                        ;4518   ;Instead of that microinstruction, insert here the code which works :
                                        ;4519
                                        ;4520   115E:   ;--------------------------------;
                                        ;4521           Q_ID[T0],                       ; GET SRC-ADDRESS
                                        ;4522           A[U_0(A),RC[T3]_ALU,            ; CLEAR OVERFLOW REGISTER
U 115E, 0003,093C,C1F0,2D98,0000,0A32   ;4523           IR2=1?,J/P2N.RES                ;
```

```
                                ;4524   ;Date    : 06-DEC-77                                                    Patch no. 028
                                ;4525   ;Author : T. FOSSUM
                                ;4526   ;
                                ;4527   ;Source code file  : IE
                                ;4528   ;First included in :
                                ;4529   ;
                                ;4530   ;PROM address to be intercepted : 011C
                                ;4531   ;
                                ;4532   ;Description of problem :
                                ;4533   ;       DURING COMPATIBILITY MODE TRAPS AND INTERRUPTS,
                                ;4534   ;       THE HIGH 16 BITS OF THE VA MAY BE INHIBITED, THUS
                                ;4535   ;       CAUSING RANDOM BEHAVIOUR IF THE SCB IS NOT LOCATED
                                ;4536   ;       IN FIRST 64K OF PHYSICAL MEMORY.
                                ;4537   ;
                                ;4538   ;
                                ;4539   ;Instead of that microinstruction, insert here the code which works :
                                ;4540
                                ;4541   1160:   ;--------------------------------;
                                ;4542           TRAP.ACC[1],                    ; TRAP ACCELARATOR
                                ;4543           SC_D.AND.K[.3],                 ;
                                ;4544           MSC/INH.CM.ADDR,                ; DON'T INHIBIT HIGH 16 BITS
U 1160, 0019,00B4,0C80,F800,3C82,0DED   ;4545           J/IE.9                  ;
```

```
                                ;4546   ;Date    : 14-DEC-77                                                    Patch no. 029
                                ;4547   ;Author : T. FOSSUM
                                ;4548   ;
                                ;4549   ;Source code file  : ARITH
                                ;4550   ;First included in :
                                ;4551   ;
                                ;4552   ;PROM address to be intercepted : 02B7
                                ;4553   ;
                                ;4554   ;Description of problem :
                                ;4555   ;       EMUL-INSTRUCTION CALLS THE GENERAL INTEGER MULTIPLY ROUTINE
                                ;4556   ;       WHICH DOES NOT WORK IF MULTIPLICAND IS MOST NEGATIVE
                                ;4557   ;       NUMBER. THUS, FOR EXAMPLE, USING EMUL TO MULTIPLY
                                ;4558   ;       6 BY 80000000 GIVES INCORRECT RESULT.
                                ;4559   ;
                                ;4560   ;HOW CAN THIS ECO BE TESTED?
                                ;4561   ;       RUN EMUL-DIAGNOSTICS.
                                ;4562   ;
                                ;4563   ;Instead of that microinstruction, insert here the code which works :
                                ;4564
                                ;4565   1161:   ;-------------------------------;
                                ;4566           ALU_RC[TO],CLK.UBCC,            ; CLOCK 2*M'CAND
U 1161, 0F10,0038,61E0,F900,0094,70B0   ;4567   Q_D,D_0,SC_K[.F]                 ; Q GETS MULTIPLIER, SC GETS COUNT
                                ;4568
                                ;4569           ;-------------------------------;
                                ;4570           LAB_R[R15],                     ; LB GETS MULTIPLICAND
                                ;4571           D_D.RIGHT,SI/ASHR,              ; D GETS LOW BIT OF Q
U 10B0, 0600,013C,0080,FA78,0000,1098   ;4572   Z?                              ; TEST FOR MOST NEGATIVE NUMBER
                                ;4573   =0
                                ;4574           ;0--------------------------------; BRANCH ON ALU Z-BIT
U 1098, 0C00,003C,01F8,F800,0000,01C0   ;4575   Q_0,D_Q,J/AR.PA.29               ; NO, REJOIN ORIGINAL ROUTINE
                                ;4576
                                ;4577           ;1--------------------------------;
                                ;4578           ALU_Q,Q_ALU.RIGHT,              ; DUPLICATE HIGH BIT OF Q
U 1099, 0041,2D3C,00C0,F800,0000,1036   ;4579   SI/ASHR,D31?                     ; Q31 GETS Q31, TEST Q0
                                ;4580   =110
                                ;4581           ;110------------------------------; BRANCH ON D31
                                ;4582           SC_K[.FFF0],                    ; PROBABLY NOT NECESSARY
                                ;4583           INTRPT.STROBE,                  ; STROBE INTERRUPTS
U 1036, 001F,0000,6DC0,F800,4084,62A2   ;4584   Q_0-Q,J/EMUL.2                   ; NEGATE Q
                                ;4585
                                ;4586           ;111------------------------------;
                                ;4587           SC_K[.FFF0],                    ; PROBABLY NOT NECESSARY
                                ;4588           INTRPT.STROBE,                  ; STROBE INTERRUPTS
U 1037, 0001,2028,6DC0,F800,4084,62A2   ;4589   Q_NOT.Q,J/EMUL.2                 ; 1'S COMPLEMENT Q
```

```
                                    ;4590   ;Date    : 15-DEC-77                                                  Patch no. 030
                                    ;4591   ;Author : T. FOSSUM
                                    ;4592   ;
                                    ;4593   ;Source code file  : CHAR
                                    ;4594   ;First included in :
                                    ;4595   ;
                                    ;4596   ;PROM address to be intercepted : 04F3 and 00F4
                                    ;4597   ;
                                    ;4598   ;Description of problem :
                                    ;4599   ;       SOME INTERRUPTIBLE INSTRUCTIONS DO NOT CLEAR THE TP-BIT
                                    ;4600   ;       DURING AN INTERRUPT, THUS MAY NOT PROGRESS WHILE INTERRUPTS
                                    ;4601   ;       OCCUR AND T-BIT IS SET IN PSL.
                                    ;4602   ;
                                    ;4603   ;Instead of that microinstruction, insert here the code which works :
                                    ;4604
                                    ;4605   1162:   ;--------------------------------;
                                    ;4606           Q_ID[PSL],                      ; GET CURRENT PSL
                                    ;4607           ALU_-1,D_ALU.RIGHT2,SI/ZERO,    ; BUILD MASK FOR BITS 29:0
U 1162, 0883,0028,3DF0,2C00,0000,09A9   ;4608           J/AS.PA.30                      ; JOIN INTERRUPT HANDLER
```

```
                                    ;4609   ;Date    : 19-DEC-77                                                Patch no. 031
                                    ;4610   ;Author : T. FOSSUM
                                    ;4611   ;
                                    ;4612   ;Source code file  : FLOAT
                                    ;4613   ;First included in :
                                    ;4614   ;
                                    ;4615   ;PROM address to be intercepted : 0428
                                    ;4616   ;
                                    ;4617   ;Description of problem :
                                    ;4618   ;       IN ADDD-INSTRUCTION, THE CONDITION CODES ARE SET ON THE
                                    ;4619   ;       SOURCE RATHER THAN THE DESTINATION WHEN THE SRC IS 0 WITH
                                    ;4620   ;       RESPECT TO THE DESTINATION.
                                    ;4621   ;
                                    ;4622   ;HOW CAN THIS ECO BE TESTED?
                                    ;4623   ;       RUN ADDD DIAGNOSTICS WITH A CASE AS DESCRIBED ABOVE.
                                    ;4624   ;
                                    ;4625   ;Instead of that microinstruction, insert here the code which works :
                                    ;4626
                                    ;4627   1163:   ;-------------------------------;
                                    ;4628           RC[T1]_Q,                       ; STORE DST<L> IN RESULT LOW
U 1163, 0001,203C,0180,F988,0000,05B2 ;4629         J/ADDD.8                        ; JUMP TO ADD.8
```

```
                                    ;4630   ;Date    : 14-FEB-78                                                    Patch no. 032
                                    ;4631   ;Author : R. LARY
                                    ;4632   ;
                                    ;4633   ;Source code file  : CHAR
                                    ;4634   ;First included in :
                                    ;4635   ;
                                    ;4636   ;PROM address to be intercepted : 0A0C
                                    ;4637   ;
                                    ;4638   ;Description of problem :
                                    ;4639   ;       BACKWARDS MOVC INSTRUCTIONS WITH COUNTS OF 256N+1,2 OR 3
                                    ;4640   ;       MAY TERMINATE PREMATURELY DUE TO BYTE-MODE SETTING OF UBCC CODES.
                                    ;4641   ;
                                    ;4642   ;HOW CAN THIS ECO BE TESTED?
                                    ;4643   ;       MOVC3 #257.,A,A+1 WHERE A IS LWORD-ALIGNED SHOULD MOVE MORE THAN 1 BYTE
                                    ;4644   ;
                                    ;4645   ;Instead of that microinstruction, insert here the code which works :
                                    ;4646
                                    ;4647   1164:
                                    ;4648   .MOVCRDBCK1:
                                    ;4649           ;-------------------------------;
                                    ;4650           D[BYTE]_CACHE, LA_RA[R3],       ; READ A BYTE
U 1164, 0000,803C,0580,4098,1404,30B4 ;4651         STATE_STATE.OR.K[.1]            ; SET BYTE OPERATION FLAG
                                    ;4652
                                    ;4653           ;-------------------------------;
                                    ;4654           Q_Q-K[.1], CLK.UBCC,            ; DECREMENT COUNT BY 1,
U 10B4, 0019,2E00,05C0,F800,0010,07B6 ;4655         INTERRUPT.REQ?, J/MOVCBCKWRITE  ; GO WRITE THE BYTE
```

```
                                    ;4656   ;Date    : 14-FEB-78                                                    Patch no. 033
                                    ;4657   ;Author : R. LARY
                                    ;4658   ;
                                    ;4659   ;Source code file  : CHAR
                                    ;4660   ;First included in :
                                    ;4661   ;
                                    ;4662   ;PROM address to be intercepted : 0A06
                                    ;4663   ;
                                    ;4664   ;Description of problem :
                                    ;4665   ;       FORWARD MOVC INSTRUCTIONS WITH COUNTS OF 256N+1,2 OR 3
                                    ;4666   ;       MAY TERMINATE PREMATURELY DUE TO BYTE-MODE SETTING OF UBCC CODES.
                                    ;4667   ;
                                    ;4668   ;HOW CAN THIS ECO BE TESTED?
                                    ;4669   ; MOVC3 #257.,A,A-1 WHERE A IS LONGWORD-ALIGNED SHOULD MOVE MORE THAN 1 BYTE.
                                    ;4670   ;
                                    ;4671   ;Instead of that microinstruction, insert here the code which works :
                                    ;4672
                                    ;4673   1165:
                                    ;4674   .MOVCRDBYTE:
                                    ;4675           ;--------------------------------;
                                    ;4676           D[BYTE]_CACHE,                   ; READ A BYTE
U 1165, 0000,803C,0580,4000,1404,30C2 ;4677         STATE_STATE.OR.K[.1]             ; SET BYTE OPERATION FLAG
                                    ;4678
                                    ;4679           ;--------------------------------;
                                    ;4680           Q_Q-K[.1], CLK.UBCC,             ; DECREMENT COUNT BY 1,
U 10C2, 0019,2000,05C0,F800,0010,06CF ;4681         J7MOVCFWRITE                     ; GO WRITE THE BYTE
```

```
                                    ;4682   ;Date    : 16-FEB-78                                                    Patch no. 034
                                    ;4683   ;Author : T. FOSSUM
                                    ;4684   ;
                                    ;4685   ;Source code file  : ARITH
                                    ;4686   ;First included in :
                                    ;4687   ;
                                    ;4688   ;PROM address to be intercepted : 0374
                                    ;4689   ;
                                    ;4690   ;Description of problem :
                                    ;4691   ;       DIV DOES NOT REPLACE QUOTIENT WITH DIVIDEND WHEN
                                    ;4692   ;       DIVISOR IS ZERO.
                                    ;4693   ;
                                    ;4694   ;HOW CAN THIS ECO BE TESTED?
                                    ;4695   ;       DIVIDE BY ZERO.
                                    ;4696   ;
                                    ;4697   ;Instead of that microinstruction, insert here the code which works :
                                    ;4698
                                    ;4699   1166:   ;-------------------------------;
                                    ;4700           ALU_D,N&Z_ALU.V&C_0,            ; CLOCK CONDITION CODES
                                    ;4701           DT/INST.DEP,                    ; BYTE, WORD OR LONG
U 1166, 0001,C03E,01E0,F800,0050,0001 ;4702         Q_D,RETURN1                     ; CRUX OF ECO
```

```
                                ;4703   ;Date    : 27-FEB-78                                                    Patch no. 035
                                ;4704   ;Author : JUD LEONARD
                                ;4705   ;
                                ;4706   ;Source code file  : FLOAT
                                ;4707   ;First included in :
                                ;4708   ;
                                ;4709   ;PROM address to be intercepted : 0E60
                                ;4710   ;
                                ;4711   ;Description of problem :
                                ;4712   ;       COMPATABILITY MODE ACCESS VIOLATIONS, WHEN THE PAGE IS ALREADY
                                ;4713   ;       MAPPED BY THE TRANSLATION BUFFER, RETURN THE 32-BIT CONTENTS OF
                                ;4714   ;       VA RATHER THAN ZERO-EXTENDING IT.  THIS ESSENTIALLY REQUIRES
                                ;4715   ;       A READ-ONLY PAGE TO BE READ THEN WRITTEN FROM COMPATABILITY MODE.
                                ;4716   ;
                                ;4717   ;HOW CAN THIS ECO BE TESTED?
                                ;4718   ;       RUN DZ11 DIAGNOSTIC (DZDZA) UNDER DIAGNOSTIC SUPERVISOR.
                                ;4719   ;
                                ;4720   ;Instead of that microinstruction, insert here the code which works :
                                ;4721
                                ;4722   1167:   ;-------------------------------;
                                ;4723           RC[VA.SV]_PC,                   ; SAVE VA'S ENTRY CONTENTS
                                ;4724           ID[Q.SV]_D,                     ; SAVE ENTRY Q REG
U 1167, 0814,1C38,BD80,3DC8,0000,105C   ;4725   D_PC,PSL.MODE?                   ; COPY VA, TEST FOR COMPATABILITY MODE
                                ;4726   =11100
                                ;4727           ;11100--------------------------; COMPATABILITY MODE
                                ;4728           RC[VA.REF]_D.0XT[WORD],         ; REPORT 16-BIT COMPATABILITY MODE ADDR
U 105C, 0003,403C,0180,F9F0,0000,0E62   ;4729   J/MM.PA.35                      ;
                                ;4730
                                ;4731           ;11101--------------------------; NOT COMPATABILITY MODE
                                ;4732           RC[VA.REF]_PC,                  ; REPORT FULL 32-BIT ADDRESS
U 105D, 0014,0038,0180,F9F0,0000,0E62   ;4733   J/MM.PA.35                      ;
                                ;4734
                                ;4735           ;11110--------------------------; NOT COMPATABILITY MODE
                                ;4736           RC[VA.REF]_PC,                  ; REPORT FULL 32-BIT ADDRESS
U 105E, 0014,0038,0180,F9F0,0000,0E62   ;4737   J/MM.PA.35                      ;
                                ;4738
                                ;4739           ;11111--------------------------; NOT COMPATABILITY MODE
                                ;4740           RC[VA.REF]_PC,                  ; REPORT FULL 32-BIT ADDRESS
U 105F, 0014,0038,0180,F9F0,0000,0E62   ;4741   J/MM.PA.35                      ;
                                ;4742   =
```

```
                                    ;4743   ;Date    : 17-MAR-78                                                    Patch no. 036
                                    ;4744   ;Author : Jud Leonard
                                    ;4745   ;
                                    ;4746   ;Source code file  : FIELD
                                    ;4747   ;First included in :
                                    ;4748   ;
                                    ;4749   ;PROM address to be intercepted : 09D0
                                    ;4750   ;
                                    ;4751   ;Description of problem :
                                    ;4752   ;       INSV touches the next longword when an inserted field includes
                                    ;4753   ;       bit 31 of an aligned longword.  ie, if P (position within the
                                    ;4754   ;       aligned longword) + S (field size) = 32, INSV touches two
                                    ;4755   ;       longwords instead of just one.
                                    ;4756   ;
                                    ;4757   ;HOW CAN THIS ECO BE TESTED?
                                    ;4758   ;       INSV R0, #24, #8, PAGE+508
                                    ;4759   ;               where PAGE is writable and PAGE+512 is not.
                                    ;4760   ;               Prior to ECO this will get access violation, should not after.
                                    ;4761   ;
                                    ;4762   ;Instead of that microinstruction, insert here the code which works :
                                    ;4763
                                    ;4764   1170:   ;-------------------------------;
                                    ;4765           Q_0+MASK+1,                     ; ZEROS TO RIGHT OF FIELD
                                    ;4766           RC[T4]_ALU,                     ; KEEP HANDY
                                    ;4767           SC_FE,                          ; GET P+S-32 TO SC
                                    ;4768           FE_SC-K[.20],                   ; P-32 TO FE
                                    ;4769           D[LONG]_CACHE.WCHK,             ; GET FIELD
U 1170, 0003,1210,75C0,51A0,0185,B0B2 ;4770         EALU?                           ; DOES IT CROSS LONGWORD BOUNDARY?
                                    ;4771   =001*
                                    ;4772           ;001*-------------------------; EALU N=0, Z=0, FIELD CROSSES BOUNDARY
                                    ;4773           RC[T3]_D.ANDNOT.Q,              ; SAVE LOW PART OF MEMORY DATA
                                    ;4774           Q_ID[T0],                       ; GET DATA TO BE INSERTED
                                    ;4775           SC_FE,                          ; GET P-32 FOR ALIGNING INSERT DATA
                                    ;4776           FE_SC,                          ; SAVE P+S-32 (WHICH IS POSITIVE)
                                    ;4777           VA_VA+4,                        ; ADDR OF SECOND LONGWORD
U 10B2, 001D,0024,C1F0,2D9B,0181,09D5 ;4778         J/INSV.6
                                    ;4779
                                    ;4780           ;011*-------------------------; EALU Z=1, FIELD ABUTS BOUNDARY
                                    ;4781           RC[T4]_Q,                       ; MASK FOR BITS OF MEMORY TO DISCARD
                                    ;4782           SC_FE,                          ; GET P-32 FOR SHIFT
U 10B6, 0001,203C,C1F0,2DA0,0081,09D1 ;4783         Q_ID[T0],J/FI.PA.36             ; GET DATA TO INSERT
                                    ;4784
                                    ;4785           ;101*-------------------------; EALU N=1, FIELD IS IN ONE LONGWORD
                                    ;4786           RC[T4]_Q-MASK-1,                ; MASK FOR BITS OF MEMORY TO DISCARD
                                    ;4787           SC_FE,                          ; GET P-32 FOR SHIFT
U 10BA, 0001,2008,C1F0,2DA0,0081,09D1 ;4788         Q_ID[T0],J/FI.PA.36             ; GET DATA TO INSERT
                                    ;4789   =;END OF EALU TEST
```

```
                                        ;4790   ;Date    : 11-APR-78                                                    Patch no. 037
                                        ;4791   ;Author : JUD LEONARD
                                        ;4792   ;
                                        ;4793   ;Source code file  : MOVPR
                                        ;4794   ;First included in :
                                        ;4795   ;
                                        ;4796   ;PROM address to be intercepted : 0D5B
                                        ;4797   ;
                                        ;4798   ;Description of problem :
                                        ;4799   ;       MTPR TO THE PAGE TABLE LENGTH REGISTERS SHOULD ACCEPT THE CONTENTS
                                        ;4800   ;       OF THE CORRESPONDING LOCATIONS IN THE PCB, WHICH HAVE OTHER QUANTITIES
                                        ;4801   ;       IN BITS 31:24.  THUS THE MBZ TEST IN THE MOVPR CODE IS WRONG,
                                        ;4802   ;       AND MUST IGNORE BITS 31:24.
                                        ;4803   ;
                                        ;4804   ;HOW CAN THIS ECO BE TESTED?
                                        ;4805   ;       MTPR #FF000000,#P0LR    ;SHOULD NOT TAKE EXCEPTION
                                        ;4806   ;
                                        ;4807   ;Instead of that microinstruction, insert here the code which works :
                                        ;4808
                                        ;4809   1171:   ;--------------------------------;
U 1171, 0000,003C,7D80,F800,0084,10C9   ;4810           SC_K[.18]                       ; READY TO BUILD MASK OF BIT 24
                                        ;4811
                                        ;4812           ;--------------------------------;
                                        ;4813           RC[T1]_0+MASK+1,                ; MASK FOR BITS 31:24
U 10C9, 0003,0010,0180,F988,0081,10CA   ;4814           SC_FE                           ; RESTORE REGISTER #
                                        ;4815
                                        ;4816           ;--------------------------------;
                                        ;4817           Q_Q.ANDNOT.RC[T1],              ; CLEAR 31-24
                                        ;4818           D_D.SWAP,                       ; THEN DO WHAT ECO'D LOC'N DID
                                        ;4819           SC_SC+K[.10],                   ;
U 10CA, 0B11,2024,65C0,F908,0084,8D60   ;4820           J/MO.PA.37                      ;
```

```
                                    ;4821   ;Date    : 11-APR-78                                                    Patch no. 038
                                    ;4822   ;Author : JUD LEONARD
                                    ;4823   ;
                                    ;4824   ;Source code file  : MOVPR
                                    ;4825   ;First included in :
                                    ;4826   ;
                                    ;4827   ;PROM address to be intercepted : 08BC
                                    ;4828   ;
                                    ;4829   ;Description of problem :
                                    ;4830   ;       MBZ CHECK ON VALUE LOADED INTO PROCESS AND SYSTEM BASE REGISTERS
                                    ;4831   ;       IS INCORRECTLY CODED, SO IT TESTS BITS 26 AND (SBR ONLY) 27 RATHER
                                    ;4832   ;       THAN 30 AND (SBR ONLY) 31 FOR ZERO.  CASE WHICH SHOWED UP THE
                                    ;4833   ;       ERROR WAS:
                                    ;4834   ;               MTPR #84000000,#P1BR
                                    ;4835   ;
                                    ;4836   ;HOW CAN THIS ECO BE TESTED? USE MTPR ABOVE
                                    ;4837   ;
                                    ;4838   ;Instead of that microinstruction, insert here the code which works :
                                    ;4839
                                    ;4840   1172:   ;-------------------------------;
U 1172, 0000,003C,0180,F800,0000,08BD ;4841         J/MO.PA.38                      ; IF TEST FAILS, PROCEED AS IF SUCCESS
```

```
                                ;4842   .TOC    ''      PCS microcode patches : Patches first included in WCS116''
                                ;4843
                                ;4844   ;Date   : 21-JUN-78                                                  Patch no. 039
                                ;4845   ;Author : Jud Leonard
                                ;4846   ;
                                ;4847   ;Source code file  : DECMAL
                                ;4848   ;First included in : WCS116
                                ;4849   ;
                                ;4850   ;PROM address to be intercepted : 0C54
                                ;4851   ;
                                ;4852   ;Description of problem :
                                ;4853   ;       ASHP gives result too large by 10**7 if rounding causes
                                ;4854   ;       a carry to propagate out of the least significant digit
                                ;4855   ;       of the result.
                                ;4856   ;       Case:   ASHP    #-1,#2,SRC,#5,#8,DST    ;Shift SRC right with rounding
                                ;4857   ;       SRC:    .BYTE   09,9C                   ;+99
                                ;4858   ;       DST:    .BYTE   0,0,0,0,0               ;Space for result
                                ;4859   ;
                                ;4860   ;HOW CAN THIS ECO BE TESTED? Deposit and run the following:
                                ;4861   ; 100/  02FF8FF8
                                ;4862   ; 104/  080506AF
                                ;4863   ; 108/  000005AF
                                ;4864   ; 10C/  00009C09
                                ;4865   ; 110/  0
                                ;4866   ;       Start at 100.  It will halt at 10B.  If the byte at 10F is zero,
                                ;4867   ;       the bug is fixed.  If one, the bug still exists.
                                ;4868   ;
                                ;4869   ;Instead of that microinstruction, insert here the code which works :
                                ;4870
                                ;4871   1173:
                                ;4872   .NEG.4: ;-------------------------------;
U 1173, 0019,2034,61C0,F800,0000,10D3   ;4873           Q_Q.AND.K[.F]                   ; STRIP GARBAGE FROM ROUND DIGIT
                                ;4874
                                ;4875           ;-------------------------------;
                                ;4876           D_D+Q,                          ; ADD ROUNDING OPERAND TO SHIFTED RESULT
                                ;4877           Q_DEC.CON,                      ; PREPARE FOR DECIMAL CORRECTION
                                ;4878           SET.CC(LONG),                   ; CATCH CARRY FOR HIGHER DIGITS
U 10D3, 081D,0014,01D0,F800,0070,064A   ;4879           J/NEG.3                         ;
```

```
                                    ;4880   ;Date    : 21-JUN-78                                                Patch no. 040
                                    ;4881   ;Author : Jud Leonard
                                    ;4882   ;
                                    ;4883   ;Source code file  : DECMAL
                                    ;4884   ;First included in : WCS116
                                    ;4885   ;
                                    ;4886   ;PROM address to be intercepted : 093E
                                    ;4887   ;
                                    ;4888   ;Description of problem :
                                    ;4889   ;               CVTTP stores the wrong thing when src len is one.
                                    ;4890   ;       Case:   CVTTP #1,SRC,TBL,#1,DST
                                    ;4891   ;       SRC:    .BYTE   0
                                    ;4892   ;       DST:    .BYTE   0
                                    ;4893   ;       TBL:    .BYTE   ^X5E
                                    ;4894   ;
                                    ;4895   ;HOW CAN THIS ECO BE TESTED? Deposit and run the following program:
                                    ;4896   ;        100/   06AF0126
                                    ;4897   ;        104/   AF0106AF
                                    ;4898   ;        108/   00000002
                                    ;4899   ;        10C/   0000005E
                                    ;4900   ;       Start at 100.  Halt will occur at 10A.  If 10B contains 5C
                                    ;4901   ;       the problem is fixed, otherwise not.
                                    ;4902   ;
                                    ;4903   ;Instead of that microinstruction, insert here the code which works :
                                    ;4904
                                    ;4905   1174:
                                    ;4906   .T2P.6: ;-------------------------------;
                                    ;4907           STATE_STATE.OR.K[.10],          ;
                                    ;4908           D_D.SWAP,                       ;
                                    ;4909           Q_R[R2],CLK.UBCC,               ;
U 1174, 0B00,003C,65C0,FA10,1414,2197   ;4910           J7T2P.LONG1                     ;
```

```
                                ;4911   .TOC    ""      PCS microcode patches : Patches first included in WCS117""
                                ;4912
                                ;4913   ;Date    : 11-SEP-78                                                  Patch no. 041
                                ;4914   ;Author : Jud Leonard
                                ;4915   ;
                                ;4916   ;Source code file  : EDIT
                                ;4917   ;First included in : WCS117
                                ;4918   ;
                                ;4919   ;PRUM address to be intercepted : 0837
                                ;4920   ;
                                ;4921   ;Description of problem :
                                ;4922   ;       The SRM definition of the EDIT operator REPLACE_SIGN is changed to
                                ;4923   ;       perform the replacement if PSW<Z> is set, and ignore PSW<N>.
                                ;4924   ;
                                ;4925   ;HOW CAN THIS ECO BE TESTED?
                                ;4926   ;       Deposit the following and start at 100:
                                ;4927   ;       100/    088F0138
                                ;4928   ;       104/    088F078F
                                ;4929   ;       108/    0
                                ;4930   ;       10C/    0001460C
                                ;4931   ;       110/    0
                                ;4932   ;       It should halt at 109.  Before the ECO, 111 would contain 0; afterwards,
                                ;4933   ;       111 should contain 20.
                                ;4934   ;
                                ;4935   ;Instead of that microinstruction, insert here the code which works :
                                ;4936
                                ;4937   1175:   ;-------------------------------;
                                ;4938           Q_0,                            ; N=0, Z=1: COPY N=Z=1 CASE
                                ;4939           STATE_STATE.OR.PATT2,           ;
                                ;4940           CACHE_D[BYTE],                  ;
U 1175, 0000,803C,09F8,3000,1404,21C4   ;4941           J/EDITNEXT                      ;
```

```
                                    ;4942   ;Date   : 11-SEP-78                                                    Patch no. 042
                                    ;4943   ;Author : Jud Leonard
                                    ;4944   ;
                                    ;4945   ;Source code file  : MMFW
                                    ;4946   ;First included in : WCS117
                                    ;4947   ;
                                    ;4948   ;PROM address to be intercepted : 017B
                                    ;4949   ;
                                    ;4950   ;Description of problem :
                                    ;4951   ;       If an unaligned read falls across a page boundary, and the higher-
                                    ;4952   ;       address page is known to the TB but inaccessible to the current mode,
                                    ;4953   ;       PC reported on the resulting fault is incorrect.
                                    ;4954   ;
                                    ;4955   ;Instead of that microinstruction, insert here the code which works :
                                    ;4956
                                    ;4957   ;=00
                                    ;4958   1168:   ;00-----------------------------;
                                    ;4959           MCT/MEM.NOP,                    ;
U 1168, 0000,1C3D,0180,0800,0000,023C ;4960         CALL,PSL.MODE?,J/GET.PTE        ;
                                    ;4961
                                    ;4962   1169:   ;01-----------------------------;
                                    ;4963           TEST.TB.RCHK,                   ;
U 1169, 0000,003C,0180,0000,0000,0D5C ;4964         J/PROBE.VA                      ;
                                    ;4965
                                    ;4966   116A:   ;10-----------------------------;
                                    ;4967           MCT/MEM.NOP,                    ;
U 116A, 0000,113C,0180,0800,0000,0D7C ;4968         LAST.REF?,J/M.FLT               ;
```

```
                                    ;4969   ;Date   : 11-SEP-78                                                    Patch no. 043
                                    ;4970   ;Author : Jud Leonard
                                    ;4971   ;
                                    ;4972   ;Source code file  : IE
                                    ;4973   ;First included in : WCS117
                                    ;4974   ;
                                    ;4975   ;PROM address to be intercepted : 0F95
                                    ;4976   ;
                                    ;4977   ;Description of problem :
                                    ;4978   ;       An SRM change redefines behaviour of exceptions when vector selects
                                    ;4979   ;       service on the interrupt stack.  They now raise IPL to 1F.
                                    ;4980   ;       NOTE 2 WORD PATCH
                                    ;4981   ;
                                    ;4982   ;Instead of that microinstruction, insert here the code which works :
                                    ;4983
                                    ;4984   116B:   ;--------------------------------;
                                    ;4985           R[SP]&VA_LA-K[.4],              ; already on IS, update it for push
U 116B, 0018,1700,1180,FAF0,0200,10D9 ;4986         STATE1?,J/IE.13B                ; distinguish except from interrupt
```

```
                                ;4987   ;Date    : 11-SEP-78                                                    Patch no. 044
                                ;4988   ;Author  : Jud Leonard
                                ;4989   ;
                                ;4990   ;Source code file  : IE
                                ;4991   ;First included in : WCS117
                                ;4992   ;
                                ;4993   ;PROM address to be intercepted : 0CED
                                ;4994   ;
                                ;4995   ;Description of problem :
                                ;4996   ;       CONTINUATION OF CHANGE TO SET IPL TO 1F ON EXCEPTION SELECTING IS
                                ;4997   ;
                                ;4998   ;Instead of that microinstruction, insert here the code which works :
                                ;4999
                                ;5000   116C:   ;-------------------------------;
                                ;5001           Q_ID[ISP],                      ; select IS, get it
                                ;5002           D_RC[T2],                       ; get old PSL for pushing on stack
U 116C, 0810,1738,B1F0,2D10,0000,1049   ;5003           STATE1?                         ;
                                ;5004   =1*01
                                ;5005           ;1*01---------------------------; got here on an exception
                                ;5006           R[SP]&VA_Q-K[.4],               ; set up decremented stack address
U 1049, 0019,2000,1180,FAF0,0200,10D9   ;5007           J/IE.13B                        ; go set IPL to 1F
                                ;5008
                                ;5009           ;1*11---------------------------; got here on interrupt
                                ;5010           INTRPT.ACK,                     ; acknowledge interrupt service
                                ;5011           R[SP]&VA_Q-K[.4],               ; set up appropriate stack address
                                ;5012           SC_FE,                          ; get 26 (bit # of IS) into SC
U 104B, 0019,2000,1180,FAF0,8281,0DF1   ;5013           J/IE.13A                        ;
                                ;5014   =1*01
                                ;5015   IE.13B: ;1*01---------------------------;
                                ;5016           EXCEPT.ACK,SC_FE,               ; here exception selected IS service
                                ;5017           D_K[.1F00],                     ; prepare to force max IPL
U 10D9, 0818,0038,9180,F800,C081,10E1   ;5018           J/IE.8A                         ;
                                ;5019
                                ;5020           ;1*11---------------------------;
                                ;5021           INTRPT.ACK,                     ;
U 10DB, 0810,0038,0180,F910,8000,0B20   ;5022           D_RC[T2],J/CALLFPD              ; go serve interrupt
                                ;5023
                                ;5024   IE.8A:  ;-------------------------------;
                                ;5025           D_D.SWAP,                       ; make IPL field of 1F
U 10E1, 0B00,003C,3DF0,2C00,0000,10E2   ;5026           Q_ID[PSL]                       ; get PSL generated by EXCEPT.ACK
                                ;5027
                                ;5028           ;-------------------------------;
U 10E2, 0801,001C,0180,F800,0000,10F8   ;5029           D_D[ORNOT]MASK                  ; set IS = 1
                                ;5030
                                ;5031           ;-------------------------------;
U 10F8, 081D,0030,0180,F800,0000,10FC   ;5032           D_D.OR.Q                        ; force IS and IPL bits of PSL
                                ;5033
                                ;5034           ;-------------------------------;
                                ;5035           ID[PSL]_D,                      ; let processor use new PSL
                                ;5036           D_RC[T2],                       ; get old one to save on stack
U 10FC, 0810,0038,3D80,3D10,0000,0B20   ;5037           J/CALLFPD                       ; right after protecting from SNV err
```

```
                                ;5038   ;Date    : 11-SEP-78                                                    Patch no. 045
                                ;5039   ;Author : Jud Leonard
                                ;5040   ;
                                ;5041   ;Source code file  : FORKS
                                ;5042   ;First included in : WCS117
                                ;5043   ;
                                ;5044   ;PROM address to be intercepted : 0067
                                ;5045   ;
                                ;5046   ;Description of problem :
                                ;5047   ;       Address sources evaluated from immediate (ie, (PC)+ ) specifiers
                                ;5048   ;       at AFORK use the I-stream data rather than its address.
                                ;5049   ;       MOVAB #10,R1    ;puts 7 in R1, not address
                                ;5050   ;
                                ;5051   ;HOW CAN THIS ECO BE TESTED?
                                ;5052   ;       Deposit and start at 100:
                                ;5053   ;       100/    51108F9E
                                ;5054   ;       104/    0
                                ;5055   ;       Before the ECO, this leaves 7 in R1.  After, it leaves 102.
                                ;5056   ;
                                ;5057   ;Instead of that microinstruction, insert here the code which works :
                                ;5058
                                ;5059   116D:   ;-------------------------------;
                                ;5060           D_Q-K[SP1.CON],                 ; backup to pc before literal
U 116D, F819,2003,15F0,F847,0000,0200   ;5061           B.FORK                  ;
```

```
                                ;5062   ;Date    : 12-SEP-83                                                 Patch no. 046
                                ;5063   ;Author : Jud Leonard
                                ;5064   ;
                                ;5065   ;Source code file  : FORKS
                                ;5066   ;First included in : WCS117
                                ;5067   ;
                                ;5068   ;PROM address to be intercepted : 0387
                                ;5069   ;
                                ;5070   ;Description of problem :
                                ;5071   ;       In certain cases, use of an immediate operand as a write destination
                                ;5072   ;       causes "microcode should never get here" machine checks, and
                                ;5073   ;       in other cases causes non-repeatable behaviour.  This use is
                                ;5074   ;       documented as "unpredictable", but such behaviour can cause
                                ;5075   ;       system crashes, which is not allowed.  In particular, the following
                                ;5076   ;       will show the failure:
                                ;5077   ;               MOVL #1,(PC)+
                                ;5078   ;       NOTE 2 WORD PATCH
                                ;5079   ;
                                ;5080   ;Instead of that microinstruction, insert here the code which works :
                                ;5081
                                ;5082   116E:   ;-------------------------------;
U 116E, 0000,003C,0180,F800,0000,0001   ;5083           J/RSVMOD                        ;
```

```
                                    ;5084   ;Date    : 12-SEP-78                                                  Patch no. 047
                                    ;5085   ;Author : Jud Leonard
                                    ;5086   ;
                                    ;5087   ;Source code file  : FORKS
                                    ;5088   ;First included in : WCS117
                                    ;5089   ;
                                    ;5090   ;PROM address to be intercepted : 0319, 04D3
                                    ;5091   ;
                                    ;5092   ;Description of problem :
                                    ;5093   ;       Continuation of immediate operand as a write destination problem
                                    ;5094   ;
                                    ;5095   ;Instead of that microinstruction, insert here the code which works :
                                    ;5096
                                    ;5097   116F:   ;-------------------------------;
                                    ;5098           D_Q,Q_D,                        ;
                                    ;5099           CLR.IB.SPEC,                    ;
U 116F, DC00,083C,01E0,F800,0000,1093 ;5100         DATA.TYPE?                      ;
                                    ;5101   =011
                                    ;5102           ;011----------------------------;
U 1093, 0000,003C,0180,F800,0000,0001 ;5103         J/RSVMOD                        ;
                                    ;5104
                                    ;5105           ;111----------------------------;
U 1097, 0000,003E,0180,F800,0000,0010 ;5106         RETURN10                        ;
```

```
                                ;5107   ;Date    : 12-SEP-78                                                Patch no. 048
                                ;5108   ;Author : Jud Leonard
                                ;5109   ;
                                ;5110   ;Source code file  : CHA
                                ;5111   ;First included in : WCS 17
                                ;5112   ;
                                ;5113   ;PROM address to be intercepted : 0673
                                ;5114   ;
                                ;5115   ;Description of problem :
                                ;5116   ;       SKPC/LOCC does not allow interrupts in the inner loop.  There is
                                ;5117   ;       actually a branch at SKPLOOP2 which attempts to test for interrupts,
                                ;5118   ;       but there is no INTRPT.STROBE in the "aligned" loop.
                                ;5119   ;
                                ;5120   ;Instead of that microinstruction, insert here the code which works :
                                ;5121
                                ;5122   1176:   ;-------------------------------;
                                ;5123           R[R1]&VA_LA+K[.4],              ; bump source address to next longword
                                ;5124           INTRPT.STROBE,                  ; clock in interrupt requests
U 1176, 0018,1B14,1180,FA88,4200,0329   ;5125           ALU?,J/SKPLOOP1                 ;
```

```
                                    ;5126   ;Date   : 12-SEP-78                                                  Patch no. 049
                                    ;5127   ;Author : Jud Leonard
                                    ;5128   ;
                                    ;5129   ;Source code file  : CHAR
                                    ;5130   ;First included in : WCS117
                                    ;5131   ;
                                    ;5132   ;PROM address to be intercepted : 06B4
                                    ;5133   ;
                                    ;5134   ;Description of problem :
                                    ;5135   ;       MOVTC/MOVTUC does not allow interrupts in the inner loop.  There is
                                    ;5136   ;       actually a branch at MOVTCLP.3 which attempts to test for interrupts,
                                    ;5137   ;       but there is no INTRPT.STROBE in either the move or fill loop.
                                    ;5138   ;
                                    ;5139   ;Instead of that microinstruction, insert here the code which works :
                                    ;5140
                                    ;5141   1177:   ;-------------------------------;
                                    ;5142           LAB_R1&RC[T2]_Q-K[.1], Q_ALU,   ;
                                    ;5143           D_ALU, CLK.UBCC, INTRPT.STROBE, ; the reason for the ECO
U 1177, 0819,3600,05C0,FB10,4010,106D ;5144         STATE5?, J/.MOVTCLP.1           ;
                                    ;5145   =1101
                                    ;5146   .MOVTCLP.1:
                                    ;5147           ;1101-----------------------------; BRANCH ON STATE<5> (FILLING)
                                    ;5148           R[R1]_LA+LC, VA_ALU, ID[TO]_D,  ; INCR SRC ADDR, LOAD VA,
U 106D, 0010,0014,C180,3E88,0200,1102 ;5149         J/.MOVTCLP.2                    ; SAVE LOOP COUNT.
                                    ;5150
                                    ;5151           ;1111-----------------------------;
U 106F, 0818,0038,1D80,F898,0000,1045 ;5152         D_K[SC], LA_RA[R3], J/.MOVTCLP.3; GET FILL IN D, GO WRITE IT
                                    ;5153
                                    ;5154   .MOVTCLP.2:
                                    ;5155           ;-------------------------------;
U 1102, 0000,803C,01C0,40A0,0000,1110 ;5156         D[BYTE]_CACHE, LA_RA[R4], Q_LA  ; READ SRC BYTE, GET TABLE ADR IN Q
                                    ;5157
                                    ;5158           ;-------------------------------;
                                    ;5159           VA_D.OXT[BYTE]+Q, Q_ID[TO],     ; INDEX INTO TABLE, RESTORE LOOPCT,
U 1110, 001F,8014,C1F0,2C98,0200,1113 ;5160         LA_RA[R3]
                                    ;5161
                                    ;5162           ;-------------------------------;
U 1113, 0000,9B3C,0180,4000,0000,1045 ;5163         D[BYTE]_CACHE, IPO?             ; READ TRANSLATED BYTE, BRANCH ON OPCODE
                                    ;5164   =*101
                                    ;5165   .MOVTCLP.3:
                                    ;5166           ;*101----------------------------; BRANCH ON IRO (MOVTUC) (N ALWAYS 0)
                                    ;5167           R[R3]_LA+LC, VA_ALU, INT?,      ; MOVTC - INCR DEST ADDR,
U 1045, 0010,0_14,0180,FA98,0200,10D6 ;5168         J/.MOVTCLP.5                    ; LOAD VA AND TEST FOR INTERRUPTS
                                    ;5169
                                    ;5170           ;*111-----------------------------;
                                    ;5171           SC_D.OXT[BYTE].XOR.K[SC],       ; MOVTUC - COMPARE BYTE TO ESC CHAR
U 1047, 001B,8E20,1D80,F800,0182,10A6 ;5172         FE_SC, INT?                     ; AND CHECK FOR INTERRUPTS
                                    ;5173   =110
                                    ;5174           ;110----------------------------; BRANCH ON INTERRUPTS (PENDING)
                                    ;5175           R[R3]_LA+LC, VA_ALU, SC_FE,     ; INCR DEST ADDR & LOAD VA,
U 10A6, 0010,1414,0180,FA98,0281,0811 ;5176         SC.GT.0?, J/MOVTCLP.4           ; RESTORE ESCAPE CHAR, TEST FOR MATCH
                                    ;5177
                                    ;5178           ;111----------------------------;
                                    ;5179           R[R3]_LA,                       ;
U 10A7, 0000,003C,0183,FA98,0000,0F82 ;5180         SD_NOT.SD,J/MOVC.RDFAULT        ; INTERRUPT - LET MOVC HANDLE IT
```

```
                                    ;5181   ;                                                   Continuation of Patch no. 049
                                    ;5182   =110
                                    ;5183   .MOVTCLP.5:
                                    ;5184           ;110----------------------------; BRANCH ON INTERRUPTS (PENDING) (MOVTC)
U 10D6, 0000,813C,0180,3000,0000,06B4 ;5185         CACHE_D[BYTE], Z?, J/MOVTCLP    ; NO INTERRUPT - WRITE BYTE & LOOP
                                    ;5186
                                    ;5187           ;111----------------------------;
                                    ;5188           R[R3]_LA,                       ;
U 10D7, 0000,003C,0183,FA98,0000,0F82 ;5189         SD_NOT.SD, J/MOVC.RDFAULT       ; INTERRUPT - JOIN COMMON CODE
```

```
                                    ;5190   ;Date   : 12-SEP-78                                                    Patch no. 050
                                    ;5191   ;Author : Jud Leonard
                                    ;5192   ;
                                    ;5193   ;Source code file  : FORKS
                                    ;5194   ;First included in : WCS117
                                    ;5195   ;
                                    ;5196   ;PROM address to be intercepted : 0008
                                    ;5197   ;
                                    ;5198   ;Description of problem :
                                    ;5199   ;       In certain pathalogical cases of instruction sequences, many memory
                                    ;5200   ;       references fall across page boundaries and the translation buffer
                                    ;5201   ;       must be updated for each reference.  Such cases can cause very long
                                    ;5202   ;       interrupt latencies.  This ECO tests for interrupts on each of the
                                    ;5203   ;       indirect addressing paths in each of the specifier evaluation
                                    ;5204   ;       routines, thus drastically shortening the worst case paths.
                                    ;5205   ;       Note : Intercepts 6 PCS addresses
                                    ;5206   ;
                                    ;5207   ;Instead of that microinstruction, insert here the code which works :
                                    ;5208
                                    ;5209   1178:   ;-------------------------------;
                                    ;5210           R(PRN)_D+K[.4].RLOG,            ;
U 1178, 0019,0018,1180,F8D8,4000,122:   ;5211           INTRPT.STROBE,J/A.IND           ;
```

```
                                    ;5212   ;Date    : 12-SEP-78                                                    Patch no. 051
                                    ;5213   ;Author : Jud Leonard
                                    ;5214   ;
                                    ;5215   ;Source code file  : FORKS
                                    ;5216   ;First included in : WCS117
                                    ;5217   ;
                                    ;5218   ;PROM address to be intercepted : 000F
                                    ;5219   ;
                                    ;5220   ;Description of problem :
                                    ;5221   ;       CONTINUATION OF ECO TO ALLOW INTERRUPTS AT INDIRECT REFERENCES
                                    ;5222   ;       Note : Intercepts 6 PCS addresses
                                    ;5223   ;
                                    ;5224   ;Instead of that microinstruction, insert here the code which works :
                                    ;5225
                                    ;5226   1179:   ;-------------------------------;
                                    ;5227           VA_Q+LB.PC,                     ;
                                    ;5228           CLR.IB.SPEC,                    ;
U 1179, D005,2014,0180,F800,4200,122C ;5229         INTRPT.STROBE                   ;
                                    ;5230
                                    ;5231   A.IND:  ;-------------------------------;
                                    ;5232           D[LONG]_CACHE,                  ;
U 122C, 0000,0E3C,0180,4000,0000,10E6 ;5233         INT?                            ;
                                    ;5234   =110
                                    ;5235           ;110----------------------------;
                                    ;5236           VA_D,                           ;
U 10E6, 0001,083C,0180,F800,0200,0094 ;5237         DATA.TYPE?,J/A.M                ;
                                    ;5238
                                    ;5239           ;111----------------------------,
U 10E7, 0000,003C,0180,F800,0000,04F8 ;5240         J/INT.B                         ;
```

```
                                    ;5241   ;Date    : 12-SEP-78                                                    Patch no. 052
                                    ;5242   ;Author : Jud Leonard
                                    ;5243   ;
                                    ;5244   ;Source code file  : FORKS
                                    ;5245   ;First included in : WCS117
                                    ;5246   ;
                                    ;5247   ;PROM address to be intercepted : 020B
                                    ;5248   ;
                                    ;5249   ;Description of problem :
                                    ;5250   ;       CONTINUATION OF ECO TO ALLOW INTERRUPTS AT INDIRECT REFERENCES
                                    ;5251   ;       Note : Intercepts 6 PCS addresses
                                    ;5252   ;
                                    ;5253   ;Instead of that microinstruction, insert here the code which works :
                                    ;5254
                                    ;5255   117A:   ;-------------------------------;
                                    ;5256           Q_D,VA_LA,                      ;
                                    ;5257           ID_D.SYNC,                      ;
U 117A, 0000,007C,15E0,BC00,4200,123A;5258          INTRPT.STROBE                   ;
                                    ;5259           ;-------------------------------;
                                    ;5260           D[LONG]_CACHE,                  ;
                                    ;5261           R(PRN)_[A+K[.4].RLOG,           ;
U 123A, 0018,0E18,1180,40D8,0000,1116;5262          INT?,J7B.IND                    ;
```

```
                                    ;5264   ;Date    : 12-SEP-78                                                    Patch no. 053
                                    ;5265   ;Author : Jud Leonard
                                    ;5266   ;
                                    ;5267   ;Source code file  : FORKS
                                    ;5268   ;First included in : WCS117
                                    ;5269   ;
                                    ;5270   ;PROM address to be intercepted : 020F
                                    ;5271   ;
                                    ;5272   ;Description of problem :
                                    ;5273   ;       CONTINUATION OF ECO TO ALLOW INTERRUPTS AT INDIRECT REFERENCES
                                    ;5274   ;       Note : Intercepts 6 PCS addresses
                                    ;5275   ;
                                    ;5276   ;Instead of that microinstruction, insert here the code which works :
                                    ;5277
                                    ;5278   117B:   ;-------------------------------;
                                    ;5279           Q_D,VA_Q+LB.PC,                 ;
                                    ;5280           ID_D.SYNC,                      ;
                                    ;5281           CLR.IB.SPEC,                    ;
U 117B, D005,2054,15E0,BC00,4200,123D ;5282         INTRPT.STROBE                   ;
                                    ;5283
                                    ;5284           ;-------------------------------;
                                    ;5285           D[LONG]_CACHE,                  ;
U 123D, 0000,0E3C,0180,4000,0000,1116 ;5286         INT?                            ;
                                    ;5287   =110
                                    ;5288   B.IND:  ;110----------------------------;
                                    ;5289           Q&VA_D,D_Q,                     ;
U 1116, 0C01,083C,01C0,F800,0200,00D0 ;5290         DATA.TYPE?,J/B.M                ;
                                    ;5291
                                    ;5292           ;111----------------------------;
U 1117, 0000,003C,0180,F800,0000,04F8 ;5293         J/INT.B                         ;
```

```
                                    ;5294   ;Date    : 12-SEP-78                                                    Patch no. 054
                                    ;5295   ;Author : Jud Leonard
                                    ;5296   ;
                                    ;5297   ;Source code file  : FORKS
                                    ;5298   ;First included in : WCS117
                                    ;5299   ;
                                    ;5300   ;PROM address to be intercepted : 030B
                                    ;5301   ;
                                    ;5302   ;Description of problem :
                                    ;5303   ;       CONTINUATION OF ECO TO ALLOW INTERRUPTS AT INDIRECT REFERENCES
                                    ;5304   ;       Note : Intercepts 6 PCS addresses
                                    ;5305   ;
                                    ;5306   ;Instead of that microinstruction, insert here the code which works :
                                    ;5307
                                    ;5308   117C:   ;-------------------------------;
                                    ;5309           Q_D,VA_LA,                      ;
U 117C, 0000,003C,01E0,F800,4200,1245;5310          INTRPT.STROBE                   ;
                                    ;5311
                                    ;5312           ;-------------------------------;
                                    ;5313           D[LONG]_CACHE,                  ;
                                    ;5314           R(PRN)_[A+K[.4].RLOG,           ;
U 1245, 0018,0E18,1180,40D8,0000,1226;5315          INT?,J7C.IND                    ;
```

```
                                        ;5316   ;Date   : 12-SEP-78                                                    Patch no. 055
                                        ;5317   ;Author : Jud Leonard
                                        ;5318   ;
                                        ;5319   ;Source code file  : FORKS
                                        ;5320   ;First included in : WCS117
                                        ;5321   ;
                                        ;5322   ;PROM address to be intercepted : 030F
                                        ;5323   ;
                                        ;5324   ;Description of problem :
                                        ;5325   ;       CONTINUATION OF ECO TO ALLOW INTERRUPTS AT INDIRECT REFERENCES
                                        ;5326   ;       Note : Intercepts 6 PCS addresses
                                        ;5327   ;
                                        ;5328   ;Instead of that microinstruction, insert here the code which works :
                                        ;5329
                                        ;5330   117D:   ;-------------------------------;
                                        ;5331           Q_D,VA_Q+LB.PC,                 ;
                                        ;5332           CLR.IB.SPEC,                    ;
U 117D, D005,2014,01E0,F800,4200,124A   ;5333           INTRPT.STROBE                   ;
                                        ;5334
                                        ;5335           ;-------------------------------;
                                        ;5336           D[LONG]_CACHE,                  ;
U 124A, 0000,0E3C,0180,4000,0000,1226   ;5337           INT?                            ;
                                        ;5338   =110
                                        ;5339   C.IND:  ;110----------------------------;
                                        ;5340           VA_D,D_Q,                       ;
U 1226, 0C01,083C,0180,F800,0200,02D2   ;5341           DATA.TYPE?,J/C.M                ;
                                        ;5342
                                        ;5343           ;111----------------------------;
U 1227, 0000,003C,0180,F800,0000,04F8   ;5344           J/INT.B                         ;
```

```
                                    ;5345   .TOC    ""      PCS microcode patches : Patches first included in WCS120""
                                    ;5346
                                    ;5347   ;Date   : 30-MAR-79                                                  Patch no. 056
                                    ;5348   ;Author : T. FOSSUM
                                    ;5349   ;
                                    ;5350   ;Source code file  : DECMAL
                                    ;5351   ;First included in : WCS120
                                    ;5352   ;
                                    ;5353   ;PROM address to be intercepted : 087D
                                    ;5354   ;
                                    ;5355   ;Description of problem :
                                    ;5356   ;       ON OVERFLOW, THE CVTPL-INSTRUCTION GIVES ONLY 31 BITS OF PRECISION.
                                    ;5357   ;       E.G.   CVTPL    #11,STRING,R0   WHERE
                                    ;5358   ;       STRING IS +33333333333, GIVES AN ANSWER WHERE BIT 31
                                    ;5359   ;       IS CLEAR WHILE IT SHOUL HAVE BEEN SET.
                                    ;5360   ;
                                    ;5361   ;Instead of that microinstruction, insert here the code which works :
                                    ;5362
                                    ;5363   1182:
                                    ;5364   PA.56.0:;-------------------------------; HERE WITH OLD SUM IN LC, NEW DATA IN D
                                    ;5365           ALU_D+LC,SHF/ALU.DT,            ; SHIFT LEFT TWICE
                                    ;5366           QK/SHF,                         ; Q GETS 4*NEW SUM
                                    ;5367           RC[T2]_ALU.LEFT2,               ; SO DOES RC2
U 1182, 0071,1214,01C0,F990,0000,1017   ;5368           EALU?                           ; TEST FOR END OF STRING
                                    ;5369   =0111
                                    ;5370           ;0111---------------------------; BRANCH ON EALU N-BIT
                                    ;5371           INTRPT.STROBE,                  ; STROBE FOR INTERRUPTS
                                    ;5372           ALU_LA+K[.1],                   ; INCREMENT SOURCE ADDRESS
                                    ;5373           R[RT]_ALU,LONG,                 ; STORE IT IN R1
                                    ;5374           VAK/LOAD,                       ; AND IN VA
                                    ;5375           QK/LEFT,D_Q,                    ; Q GETS 8*SUM, D GETS 4*SUM
J 1017, 0C18,0014,05A8,FA88,4200,125A   ;5376           J/P2LL1.P                       ; CONTINUE IN LOOP
                                    ;5377
                                    ;5378           ;1111---------------------------; THIS BYTE IS THE SIGN-BYTE
                                    ;5379           ALU_LA+K[.1],                   ;
                                    ;5380           R[RT]_ALU,LONG,                 ;
U 101F, 0018,0014,0580,FA88,0200,1254   ;5381           VAK/LOAD                        ;
                                    ;5382
                                    ;5383           ;-------------------------------;
                                    ;5384           ALU_D+LC,Q_ALU.LEFT,            ; GENERATE 2*SUM
U 1254, 0031,8014,01C0,4000,0000,1256   ;5385           D[BYTE]_CACHE                   ; READ SIGN-BYTE
                                    ;5386
                                    ;5387           ;-------------------------------;
                                    ;5388           ALU_Q,RC[T2]_ALU.LEFT2,         ; STORE 4*SUM IN RC2
U 1256, 0061,203C,0180,F990,0000,0885   ;5389           J/P2L                           ; "P2L"
```

```
                                        ;5390   ;                                                   Continuation of Patch no. 056
                                        ;5391   P2LL1.P:
                                        ;5392           ;-------------------------------;
                                        ;5393           ALU_D+Q,D_ALU.LEFT3,            ; ADD AND MULTIPLY BY 8
                                        ;5394           LC_RC[T2],                      ; LATCH SUM
U 125A, 088D,0E14,0180,F910,0000,1236   ;5395           BEN/INTERRUPT                   ;
                                        ;5396   =110
                                        ;5397           ;110----------------------------; BRANCH ON INTERRUPT REQUEST
                                        ;5398           Q_D+LC,RC[T2]_ALU,              ; SAVE SUM*100 IN RC2
U 1236, 0011,8014,01C0,4190,0000,125E   ;5399           D[BYTE]_CACHE,J/P2LL.P          ; GET NEXT INPUT BYTE
                                        ;5400
                                        ;5401           ;111----------------------------;
                                        ;5402           STATE_STATE.OR.K[.80],          ; SET INTERRUPT BIT
U 1237, 0000,003C,4180,F800,1404,2033   ;5403           J/SAVE.BCD                      ; SAVE.BCD
                                        ;5404
                                        ;5405   P2LL.P:
                                        ;5406           ;-------------------------------;
                                        ;5407           ALU_D.0XT[BYTE].AND.K[.FFF0],   ; ISOLATE HIGH NIBBLE
U 125E, 009B,8034,6DC0,F800,0000,1260   ;5408           Q_ALU.RIGHT2                    ; Q GETS NIBBLE*4
                                        ;5409
                                        ;5410           ;-------------------------------;
                                        ;5411           ALU_D.0XT[BYTE]-Q,              ; SUBTRACT 4*HIGH NIBBLE
U 1260, 081F,8000,01B0,F910,0000,1264   ;5412           D_ALU,QK/RIGHT,LC_RC[T2]        ; STORE RESULT IN D
                                        ;5413
                                        ;5414           ;-------------------------------;
                                        ;5415           ALU_D-Q,D_ALU,LA_RA[R1],        ;
                                        ;5416           SC_SC-K[.T],CLK.UBCC,           ; UPDATE SRC-LENGTH
U 1264, 081D,0000,0580,F888,0094,B182   ;5417           J/PA.56.0                       ;
```

```
                                ;5418   ;Date    : 24-MAY-79                                                    Patch no. 057
                                ;5419   ;Author : T. FOSSUM
                                ;5420   ;
                                ;5421   ;Source code file  : DECMAL
                                ;5422   ;First included in : WCS120
                                ;5423   ;
                                ;5424   ;PROM address to be intercepted : 05D6
                                ;5425   ;
                                ;5426   ;Description of problem :
                                ;5427   ;       THE CVTPS-INSTRUCTION WILL TREAT THE HIGH ORDER BIT OF THE LEADING
                                ;5428   ;       DIGIT AS OVERFLOW IF THE STRINGS ARE ALIGNED AND DST LENGTH IS EVEN.
                                ;5429   ;       CVTPS   #3,A,#2,B
                                ;5430   ;       WHERE SRC STRING IS 80, WILL SET THE V-BIT.
                                ;5431   ;
                                ;5432   ;Instead of that microinstruction, insert here the code which works :
                                ;5433
                                ;5434   1183:   ;-------------------------------;
                                ;5435           FE_SC-K[.1],SC_FE,CLK.UBCC,     ; UPDATE SRC-LENGTH
                                ;5436           ALU_LA-K[.1],                   ; UPDATE DST-ADDRESS
                                ;5437           VAK7LOAD,R[R3]_ALU,             ; STORE IT IN VA AND R3
                                ;5438           QK/RIGHT2,                      ; SHIFT LAST DIGIT RIGHT
                                ;5439           DK/RIGHT,                       ; SHIFT OVERFLOW DIGIT RIGHT ( TO GET
U 1183, 0618,0000,0590,FA98,0395,ABAA   ;5440           J/P2NL10                        ; RID OF HIGH BIT FROM LAST DIGIT)***
```

```
                                    ;5441   ;Date    : 24-MAY-79                                                    Patch no. 058
                                    ;5442   ;Author : T. Fossum
                                    ;5443   ;
                                    ;5444   ;Source code file  : FORKS
                                    ;5445   ;First included in : WCS120
                                    ;5446   ;
                                    ;5447   ;PROM address to be intercepted : 031F
                                    ;5448   ;
                                    ;5449   ;Description of problem :
                                    ;5450   ;       EMODD WILL GET RANDOM DATA IF ITS 3. OPERAND IS A LITERAL.
                                    ;5451   ;
                                    ;5452   ;Instead of that microinstruction, insert here the code which works :
                                    ;5453
                                    ;5454   1184:   ;-------------------------------;
                                    ;5455           RC[T2]_Q,                       ; QUAD IMMEDIATE ****CRUX OF ECO****
                                    ;5456           Q_IB.DATA,                      ; ORIGINAL CODE HAS RC[T1] HERE
                                    ;5457           C[r.1.3.COND,                   ; GET SECOND PART
                                    ;5458           PC_PC-4,                        ; INCREMENT PC PAST SECOND PART
U 1184, F001,2B3C,01F0,F996,0000,04D0   ;5459           IB.TEST?,J/C.IQ                 ;
```

```
                                ;5460   ;Date    : 10-JUL-79                                                Patch no. 059
                                ;5461   ;Author : T. FOSSUM, R. J. AVARBOCK(25-JUN-80)
                                ;5462   ;
                                ;5463   ;Source code file  : DECMAL
                                ;5464   ;First included in : WCS120
                                ;5465   ;
                                ;5466   ;PROM address to be intercepted : 0672
                                ;5467   ;
                                ;5468   ;Description of problem :
                                ;5469   ;       ASHP INSTRUCTON NO CHECK IS MADE ON DST-LENGTH.
                                ;5470   ;
                                ;5471   ;Instead of that microinstruction, insert here the code which works :
                                ;5472
                                ;5473   1185:   ;-------------------------------;
                                ;5474           ALU_D.AND.K[.F],                ; ISOLATE ROUND DIGIT
                                ;5475           D_ALU,                          ; SAVE IT IN D-REGISTER
                                ;5476           QK/RIGHT,                       ; GENERATE 2*SHIFT-COUNT
U 1185, 0819,1434,61B0,F800,000C.1243   ;5477           SC?                             ; TEST BITS <9:8> OF SHIFT-COUNT*4
                                ;5478   =011
                                ;5479           ;011----------------------------; BRANCH ON SC<9:8> .NE. 0
                                ;5480           D_D.SWAP,ID[T2]_D,              ; SWAP ROUND INTO HIGH BYTE & SAVE IN T2
                                ;5481           EALU_SC,CLK.UBCC,               ; CLOCK DIRECTION OF SHIFT
                                ;5482           QK/LEFT,                        ; GENERATE 4*SHIFT-COUNT
U 1243, 0B00,003C,C9A8,3C00,0010,120A   ;5483           J/ASHP.ECO.1                    ;
                                ;5484
                                ;5485           ;111----------------------------;
                                ;5486           ALU_Q.AND.K[.C0],               ; CLOCK BITS <6:5> OF SHIFT-COUNT
                                ;5487           CLK.UBCC,                       ; SET OR CLEAR ALU Z-BIT
U 1247, 0019,2034,D1A8,F800,0010,1266   ;5488           QK/LEFT,J/ASHP.ECO.2            ;
                                ;5489
                                ;5490   =0**10
                                ;5491   ASHP.ECO.1:
                                ;5492           ;0**10--------------------------;
                                ;5493           ALU_D.OR.Q,RC[T5]_ALU,          ; SAVE BOTH COUNT AND ROUND OPERANDS
U 120A, 001D,0031,0180,F9A8,0000,037E   ;5494           CALL,J/SPEC                     ; IN RC5 AND EVALUATE LENGTH OF DEST
                                ;5495   =0**11
                                ;5496           ;0**11--------------------------;
                                ;5497           ALU_Q.OR.K[.A0],Q_ALU,          ; MAKE THE COUNT SMALLER NEGATIVE NUMBER
                                ;5498           RC[T4]_ALU,SC_ALU,              ; STORE IN RC 4 AND SC
U 120B, 0019,2030,25C0,F9A0,0082,120A   ;5499           J/ASHP.ECO.1                    ;
                                ;5500   =1**10
                                ;5501           ;1**10--------------------------;
                                ;5502           ALU_D.AND.K[.FFE0],             ; CHECK LENGTH OF DEST FOR >32
U 121A, 0019,0034,A180,F800,0030,01D6   ;5503           N_AMX.Z_TST,J/DC.PA.59          ; SET CC'S RETURN TO PCS
                                ;5504   =
                                ;5505   ASHP.ECO.2:
                                ;5506           ;-------------------------------;
                                ;5507           D_D.SWAP,ID[T2]_D,              ; GET COUNT TO HIGH END OF D
                                ;5508           EALU_SC,CLK.UBCC,               ;
U 1266, 0B00,013C,C980,3C00,0010,120A   ;5509           Z?,J/ASHP.ECO.1                 ;
```

```
                                ;5510   ;Date   : 24-MAY-79                                                    Patch no. 060
                                ;5511   ;Author : R. SHORT
                                ;5512   ;
                                ;5513   ;Source code file  : FORKS
                                ;5514   ;First included in : WCS120
                                ;5515   ;
                                ;5516   ;PROM address to be intercepted : 00C7
                                ;5517   ;
                                ;5518   ;Description of problem :
                                ;5519   ;       ACB-INSTRUCTION
                                ;5520   ;       IF THE COMPARE OPERATION IN AN ACB INSTRUCTION IN WHICH THE ADDEND
                                ;5521   ;       IS POSITIVE CAUSES AN INTEGER OVERFLOW THE BRANCH IS NOT TAKEN EVEN
                                ;5522   ;       IF THE BRANCH CONDITION IS MET.
                                ;5523   ;       MOVL    #-25000,R1
                                ;5524   ;       ACBW    #25000,#5000,R1,10$             ;SHOULD BRANCH
                                ;5525   ;
                                ;5526   ;Instead of that microinstruction, insert here the code which works :
                                ;5527
                                ;5528   1186:   ;-------------------------------;
                                ;5529           ALU D-LC,                       ; COMPARE INDEX TO LIMIT
                                ;5530           DT/INST.DEP,                    ;
                                ;5531           CLK.UBCC,                       ; SAVE CC'S
U 1186, 0011,C000,0180,F800,0010,1268   ;5532           J/ACB3                          ;
                                ;5533
                                ;5534   ACB3:   ;-------------------------------;
                                ;5535           Q Q+PC,                         ; CALCULATE BRANCH OFFSET
                                ;5536           CLR.IB.OPC,                     ; THROW AWAY OPCODE
                                ;5537           PC PC+1,                        ; COUNT PC
U 1268, C015,3B14,01C0,F804,4000,10C3   ;5538           ALU.N?                          ;
                                ;5539   =0011
                                ;5540           ;0011-----------------------------; INDEX  GTR LIMIT
U 10C3, F80C,003B,01F1,F857,139B,6000   ;5541           IRD                             ; GET THE NEXT INSTRUCTION
                                ;5542
                                ;5543           ;0111-----------------------------;
                                ;5544           PC&VA Q,FLUSH.IB,               ; INDEX = LIMIT
U 10C7, 2001,203C,0180,F801,4200,00AB   ;5545           J/IB.FILL                       ; BRANCH
                                ;5546
                                ;5547           ;1011-----------------------------;
                                ;5548           PC&VA Q,FLUSH.IB,               ; INDEX LESS THEN LIMIT
U 10CB, 2001,203C,0180,F801,4200,00AB   ;5549           J/IB.FILL                       ; BRANCH
                                ;5550   =
```

```
                                    ;5551   ;Date    : 24-MAY-79                                                    Patch no. 061
                                    ;5552   ;Author : R. SHORT
                                    ;5553   ;
                                    ;5554   ;Source code file  : INDEX
                                    ;5555   ;First included in : WCS120
                                    ;5556   ;
                                    ;5557   ;PROM address to be intercepted : 0390
                                    ;5558   ;
                                    ;5559   ;Description of problem :
                                    ;5560   ;       INDEX-INSTRUCTION
                                    ;5561   ;       IF THE SIZE OPERAND IS ZERO AND THE INDEX OPERAND IS NEGATIVE
                                    ;5562   ;       THE MICROCODE HANGS UP IN AN INFINITE LOOP.
                                    ;5563   ;       R3      -100
                                    ;5564   ;       R0      -10
                                    ;5565   ;       R1      +10
                                    ;5566   ;       INDEX   R3,R0,R1,#0,#0,R2       ; HANGS UP THE MICRO CODE
                                    ;5567   ;
                                    ;5568   ;Instead of that microinstruction, insert here the code which works :
                                    ;5569
                                    ;5570   1187:   ;--------------------------------;
                                    ;5571           D_RC[T1],                       ; GET MULTIPLIER TO D (SIZE)
                                    ;5572           SC_SC+1,                        ; SC <- 100 FOR MUL ROUTINE
U 1187, 0810,0038,0180,F908,0080,C516   ;5573           J/INDEX.3                       ;
```

```
                                    ;5574   ;Date    : 25-MAY-79                                                    Patch no. 062
                                    ;5575   ;Author : R. SHORT
                                    ;5576   ;
                                    ;5577   ;Source code file  : FORKS
                                    ;5578   ;First included in : WCS120
                                    ;5579   ;
                                    ;5580   ;PROM address to be intercepted : 02EE
                                    ;5581   ;
                                    ;5582   ;Description of problem :
                                    ;5583   ;       BRANCH ON BIT INSTRUCTIONS WITH A NEGATIVE POSITION OPERAND CALCULATES
                                    ;5584   ;       THE ADDRESS OF THE BIT OPERAND INCORRECTLY.DUE TO SHIFTING IN
                                    ;5585   ;       ZEROS INSTEAD OF SIGN BIT WHEN SHIFTING POSITION RIGHT
                                    ;5586   ;       BASE:   BBC     #-1,BASE,FOO     CAUSES ACCESS VIOLATION
                                    ;5587   ;                                        EVEN IF BASE-4 IS VALID ADDRESS
                                    ;5588   ;
                                    ;5589   ;Instead of that microinstruction, insert here the code which works :
                                    ;5590
                                    ;5591   1188:   ;-------------------------------;
                                    ;5592           SC_Q,                           ; SAVE LOW BITS OF POSITION IN SC
                                    ;5593           D_Q.RIGHT2,SI/ASHR,             ; SHIFT POSITION RIGHT 2
                                    ;5594           Q_IB.BDEST,                     ; SHIFTING IN SIGN
U 1188, 7881,2B3C,00F0,F804,0082,1230   ;5595           PC_PC+1,IB.TEST?,J/.BB.1        ;
                                    ;5596   =00
                                    ;5597   .BB.1:  ;00-------------------------------;
U 1230, 0000,003D,0180,F800,0000,0E64   ;5598           CALL,J/IB.TBM                   ; TB MISS
                                    ;5599
                                    ;5600           ;01-------------------------------;
U 1231, 0000,003D,0180,F800,0000,0B80   ;5601           CALL,J/IB.ERR                   ;
                                    ;5602
                                    ;5603           ;10-------------------------------;
                                    ;5604           Q_IB.BDEST,                     ; GET THE BRANCH OFFSET TO Q
                                    ;5605           IB.TEST?,                       ; WAIT FOR THE IB TO GET IT
U 1232, 7000,083C,01F0,F800,0000,1230   ;5606           J/.BB.1                         ;
                                    ;5607
                                    ;5608           ;11-------------------------------;
                                    ;5609           CLR.IB.SPEC,                    ; THROW AWAY THE BDEST
U 1233, D85C,2038,0080,F800,0000,126A   ;5610           D_D.RIGHT(B),SI/ASHR            ; SHIFT POSITION RIGHT SHIFT IN THE
                                    ;5611                                           ; SIGN BIT
                                    ;5612
                                    ;5613           ;-------------------------------;
                                    ;5614           ALU_Q.OXT[BYTE],CLK.UBCC,       ; TEST FOR BDEST OF ZERO
                                    ;5615           LC_R[TO],                       ; READY THE BASE ADDRESS
                                    ;5616           SC_SC.ANDNOT.K[.FFF8],          ; GET THE BIT POSITION TO SC
U 126A, 0003,A03C,7180,F900,0094,42F1   ;5617           J/FO.PA.62                      ;
```

```
                                ;5618   ;Date    : 20-JUN-79                                                    Patch no. 063
                                ;5619   ;Author : R. SHORT/T. FOSSUM
                                ;5620   ;
                                ;5621   ;Source code file  : DECMAL
                                ;5622   ;First included in : WCS120
                                ;5623   ;
                                ;5624   ;PROM address to be intercepted : C8DF
                                ;5625   ;
                                ;5626   ;Description of problem :
                                ;5627   ;       COMPARE PACKED INSTRUCTION
                                ;5628   ;       IF THE SECOND STRING IS LONGER THAN THE FIRST THE Z BIT
                                ;5629   ;       IS NOT ALWAYS SET CORRECTLY
                                ;5630   ;       SRC1:   .PACKED 0
                                ;5631   ;       SRC2:   .PACKED -00
                                ;5632   ;       CMPP4   #1,SRC1,#2,SRC2         ; SHOULD SET THE Z BIT BUT DOES'NT
                                ;5633   ;
                                ;5634   ;Instead of that microinstruction, insert here the code which works :
                                ;5635
                                ;5636   1189:   ;-------------------------------;
                                ;5637           R[R3]_LA-K[.1],SC_FE,           ; UPDATE 2'ND ADDRESS
U 1189, 0F18,0000,0580,FA98,0281,08FF   ;5638           VA_ALU,D_0,J/CMP3[4             ; RESTORE THE ADDRESS
```

```
                                ;5639   ;Date   : 29-MAY-79                                                  Patch no. 064
                                ;5640   ;Author : R. SHORT
                                ;5641   ;
                                ;5642   ;Source code file  : QUEUE
                                ;5643   ;First included in : WCS120
                                ;5644   ;
                                ;5645   ;PROM address to be intercepted : OCFD
                                ;5646   ;
                                ;5647   ;Description of problem :
                                ;5648   ;       INSQUE-INSTRUCTION
                                ;5649   ;       THE CONDITION CODES ARE SET INCORRECTLY IN AN INSQUE INSTRUCTION
                                ;5650   ;       H:      .LONG   A,A
                                ;5651   ;       A:      .LONG   H,H
                                ;5652   ;       B:      .LONG   0,0
                                ;5653   ;       INSQUE  B,H SETS N AND C WRONGLY
                                ;5654   ;       ALSO THE HARDWARE CONTEXT OF THE INSTRUCTION IS BYTE CAUSEING
                                ;5655   ;       THE Z BIT TO BE SET INCORRECTLY.WHEN THE ELEMENTS ARE A MULTIPLE
                                ;5656   ;       OF 256 BYTES APART.
                                ;5657   ;
                                ;5658   ;Instead of that microinstruction, insert here the code which works :
                                ;5659
                                ;5660   118A:   ;-------------------------------;
                                ;5661           VA_VA+4,                        ; GET ADDRESS OF BLINK OF ENTRY
                                ;5662           D_Q,                            ; GET PREDECESSOR ADDRESS
                                ;5663           ALU_Q-D-1,SET.CC(LONG),         ; COMPARE PRED AND SUCC ADDRESSES
U 118A, 0C1D,2008,0180,F803,0070,0CFE   ;5664           J/QU.PA.64                      ; AND SET THE CC'S
```

```
                                ;5665   ;Date   : 30-MAY-79                                                    Patch no. C65
                                ;5666   ;Author : R. SHORT
                                ;5667   ;
                                ;5668   ;Source code file  : FIELD
                                ;5669   ;First included in : WCS120
                                ;5670   ;
                                ;5671   ;PROM address to be intercepted : 0196
                                ;5672   ;
                                ;5673   ;Description of problem :
                                ;5674   ;       INSV-INSTRUCTION
                                ;5675   ;       IF THE DESTINATION CROSSES A REGISTER BOUNDRY
                                ;5676   ;       THE HIGH PART OF THE DESTINATION FIELD IS NOT STORED CORRECTLY
                                ;5677   ;       R0 = AA
                                ;5678   ;       R1=0
                                ;5679   ;       R2=0
                                ;5680   ;       INSV    R0,#28.#8,R1 SHOULD RESULT IN
                                ;5681   ;                               R1 = A0000000
                                ;5682   ;                               R2 = 0000000A
                                ;5683   ;
                                ;5684   ;Instead of that microinstruction, insert here the code which works :
                                ;5685
                                ;5686   118B:   ;-------------------------------;
                                ;5687           D_DAL.SC,                       ; ALIGN INSERT
                                ;5688           SC_FE,FE_SC,                    ; SC<-P+S-32,SAVE P IN FE
U 118B, 0D10,0024,01C0,F920,0181,1270   ;5689           Q_[A.ANDNOT.RC[T4]              ;
                                ;5690
                                ;5691           ;-------------------------------;
U 1270, 0811,0034,0180,F800,0000,1272   ;5692           D_D.AND.LC                      ; PUT LOW PART OF INSERT INTO DATA
                                ;5693
                                ;5694           ;-------------------------------;
                                ;5695           R(PRN)_D.OR.Q,                  ; COMBINE OLD DATA WITH INSERT
U 1272, 001D,0030,C1F0,2CD8,0000,1274   ;5696           Q_ID[T0]                        ; GET BACK SOURCE DATA
                                ;5697
                                ;5698           ;-------------------------------;
                                ;5699           RC[T4]_0+MASK+1,                ; MAKE MASK FOR LOW BITS OF HIGH INSERT
U 1274, 0003,0010,0180,F9A0,0081,1276   ;5700           SC_FE                           ; GET P INTO SC
                                ;5701
                                ;5702           ;-------------------------------;
                                ;5703           D_DAL.SC,                       ; ALIGN INSERT
                                ;5704           Q_RC[T4],                       ; GET MASK TO Q
U 1276, 0D10,0038,01C0,F920,0000,09C9   ;5705           J7FI.PA.65                      ;
```

```
                                ;5706   ;Date    : 02-JUL-79                                                    Patch no. 066
                                ;5707   ;Author : T. Fossum
                                ;5708   ;
                                ;5709   ;Source code file  : DECMAL
                                ;5710   ;First included in : WCS120
                                ;5711   ;
                                ;5712   ;PROM address to be intercepted : 01EE
                                ;5713   ;
                                ;5714   ;Description of problem :
                                ;5715   ;       THE PSL Z-BIT IS NOT SET CORRECTLY IN MULP.
                                ;5716   ;       MULP    5**10. X 2**10. , #9.,A
                                ;5717   ;       CAUSES Z-BIT TO BE CLEARED, WHEN IT SHOULD NOT.
                                ;5718   ;
                                ;5719   ;Instead of that microinstruction, insert here the code which works :
                                ;5720
                                ;5721   ;THIS ROUTINE READS PRODUCT-STRING, STARTING AT THE LONGWORD
                                ;5722   ;CONTAINING THE SIGN-NIBBLE, UNTIL IT EITHER FINDS A NON-ZERO
                                ;5723   ;DIGIT, OR IT REACHES THE END OF THE STRING.
                                ;5724   ;IF A MEMORY FAULT OCCURS WHILE DOING THIS, THE ROUTINE
                                ;5725   ;WILL RESTART AT THE LABEL 'MULM', AND THEN RESTART THIS ROUTINE.
                                ;5726   ;THE STRING ADDRESS IS KEPT IN R15, THE LENGTH IN RC5.
                                ;5727   ;STATE-REGISTER IS ORED WITH 9 TO ENSURE RESTARTING.
                                ;5728
                                ;5729   118C:   ;-------------------------------;
U 118C, 0810,0038,0180,F930,0000,1001   ;5730           D_RC[T6]                        ; NEED TO ADJUST RC-COUNT
                                ;5731   =0****
                                ;5732           ;0****-------------------------;
                                ;5733           ALU_0+D+1,N&Z_ALU.V&C_0,        ; CLEAR PSL CC
                                ;5734           RC[T6]_ALU,                     ; INCREMENT RC-COUNT IN CASE OF FAULT
U 1001, 001F,2011,1180,F9B0,0[illegible]4,6BCD  ;5735           SC_K[.4], CALL, J/REG.ADJUST    ; ADJUST R5, CLEAR R4
                                ;5736
                                ;5737           ;1****-------------------------;
                                ;5738           Q_ID[T4],                       ; GET DST-LENGTH
U 1011, 0000,003C,D1F0,2E28,0000,127A   ;5739           LAB_R[R5]                       ; GET DST-ADDRESS
                                ;5740
                                ;5741           ;-------------------------------;
                                ;5742           STATE_STATE.OR.K[.9],           ; NEEDED TO CAUSE RESTART ON FAULTS
                                ;5743                                           ; TO GO TO 'MULM'
                                ;5744           ALU_Q.SXT[BYTE],D_ALU.RIGHT,    ; DIVIDE LENGTH BY 2
                                ;5745           SI/ASHR,                        ; SIGN-EXTEND
                                ;5746           RC[T5]_ALU.RIGHT,               ; USE RC5 FOR TEMPORARY
U 127A, 0842,A03C,D880,F9A8,1414,3222   ;5747           CLK.UBCC                        ; CLOCK LENGTH
                                ;5748   =10
                                ;5749           ;10------------------------------;
                                ;5750           ALU_LA-D,VAK/LOAD,              ; GENERATE ADDRESS OF SIGN-BYTE
                                ;5751           R[RT5]_ALU,                     ; USE R15 FOR ADDRESS TEMP
U 1222, 001C,2001,0180,FAF8,0200,1282   ;5752           CALL,J/MULP.ECO.2               ;
                                ;5753
                                ;5754           ;11------------------------------;
                                ;5755           ALU_D.ANDNOT.K[.F],             ; EXCLUDE SIGN-NIBBLE
                                ;5756           CLK.UBCC,                       ; CLOCK THE DATA
U 1223, 0019,0024,6180,F928,0010,127E   ;5757           LC_RC[T5]                       ; LATCH COUNT
```

```
                                        ;5758   ;                                                   Continuation of Patch no. 066
                                        ;5759   MULP.ECO.1:
                                        ;5760           ;-------------------------------;
                                        ;5761           ALU_LA-K[.4],R[R15]_ALU,        ; POINT ADDRESS TO NEXT LONGWORD
                                        ;5762           VAK?LOAD,                       ; LOAD VA WITH IT
                                        ;5763           Q_0,                            ; FOR LATER CONVENIENCE
U 127E, 0018,0100,11F8,FAF8,0200,10A4   ;5764           Z?                              ; IS DATA 0 ?
                                        ;5765
                                        ;5766   =0                                      ; BRANCH ON ALU Z-BIT
                                        ;5767           ;0-------------------------------; NON-ZERO STRING
                                        ;5768           LAB_R[R3],                      ; LATCH MUL'CAND ADDRESS
U 10A4, 0000,003C,0180,FA18,0000,1284   ;5769           J/MULP.ECO.3                    ; NON-ZERO STRING
                                        ;5770
                                        ;5771           ;1-------------------------------;
                                        ;5772           LAB_R[R15],                     ;  LATCH ADDRESS
U 10A5, 0810,0038,0180,FA78,0000,1248   ;5773           D_LC
                                        ;5774   =00
                                        ;5775           ;00------------------------------;
                                        ;5776           ALU_D+K[.4],CLK.UBCC,           ; UPDATE AND CLOCK LENGTH
                                        ;5777           RC[T5]_ALU,                     ; USE RC5 FOR STORING LENGTH
                                        ;5778           D_ALU,                          ;
U 1248, 0819,0015,1180,F9A8,0010,1282   ;5779           CALL,J/MULP.ECO.2               ;
                                        ;5780
                                        ;5781           ;01------------------------------;
                                        ;5782           ALU_0(A),N&Z_ALU.V&C_0,         ; SET PSL Z-BIT, STRING IS 0
U 1249, 0003,003C,0180,FA18,0050,1284   ;5783           LAB_R[R3],J/MULP.ECO.3          ; LATCH M'CAND ADDRESS
                                        ;5784   =11
                                        ;5785           ;11------------------------------;
                                        ;5786           ALU_D,CLK.UBCC,                 ;
U 124B, 0001,003C,0180,F928,0010,127E   ;5787           LC_RC[T5],J/MULP.ECO.1          ;
                                        ;5788
                                        ;5789   MULP.ECO.2:
                                        ;5790           ;-------------------------------;
                                        ;5791           LAB_R[R15],                     ; LATCH UP DST-ADDRESS
                                        ;5792           ALU_D+K[.3],                    ; INCREMENT ADDRESS
                                        ;5793           D_ALU,                          ;
                                        ;5794           CLK.UBCC,BYTE,                  ; CLOCK IT FOR LATER TEST
U 1282, 0819,9B14,0D80,FA78,0010,0AF7   ;5795           ALU?,J/READ0                    ;
                                        ;5796
                                        ;5797   MULP.ECO.3:
                                        ;5798           ;-------------------------------;
                                        ;5799           STATE_STATE.ANDNOT.K[.1],       ; CLEAR STATE-BIT FOR BRANCHING
U 1284, 000F,0010,0580,FA98,1404,5286   ;5800           ALU_0+LB+1,R[R3]_ALU            ; FINALIZE M'CAND ADDRESS
                                        ;5801
                                        ;5802           ;-------------------------------;
                                        ;5803           ALU_0(A),R[R0]_ALU,             ; STORE FINAL 0 IN R0
                                        ;5804           Q_ID[T6],                       ; Q GETS M'PLIER ADDRESS
                                        ;5805           STATE3-0?,                      ; TEST SIGN-BIT OF STATE
U 1286, 0003,173C,D9F0,2E80,0000,012C   ;5806           J/MULP.ECO.4                    ; RETURN TO PCS
```

```
                                        ;5807   ;Date    : 14-JUN-79                                                    Patch no. 067
                                        ;5808   ;Author : R. SHORT
                                        ;5809   ;
                                        ;5810   ;Source code file  : CORKS
                                        ;5811   ;First included in : WCS120
                                        ;5812   ;
                                        ;5813   ;PROM address to be intercepted : 01E7
                                        ;5814   ;
                                        ;5815   ;Description of problem :
                                        ;5816   ;       ASHL INSTRUCTION
                                        ;5817   ;       IF THE SHIFT IS POSITIVE AND GREATER THAN 32 THE V BIT DOES NOT GET
                                        ;5818   ;       SET IF THE SOURCE IS NON ZERO
                                        ;5819   ;       A:      .LONG   2
                                        ;5820   ;       B:      .LONG   0
                                        ;5821   ;               ASHL    #^X7F,A,B       ; SHOULD SET THE V BIT
                                        ;5822   ;
                                        ;5823   ;Instead of that microinstruction, insert here the code which works :
                                        ;5824
                                        ;5825   118D:   ;-------------------------------;
                                        ;5826           Q_0,ALU_0(A),N&Z_ALU.V&C_0,     ; RESULT IS ALWAYS ZERO,SET CCS
U 118D, 0C03,003C,01F8,F800,0050,1289   ;5827           D_Q                             ; GET SOURCE TO D FOR OVERFLOW TESTING
                                        ;5828
                                        ;5829           ;-------------------------------; IF THE SOURCE IS NON ZERO
U 1289, 0000,0D3C,0180,F800,0000,1269   ;5830           D.NE.0?                         ; THE V BIT GETS SET
                                        ;5831   =*01
                                        ;5832           ;*01----------------------------; BRANCH ON D NE 0 (Q31=0)
                                        ;5833           D_Q,                            ; NO OVERFLOW STORE RESULT
U 1269, 0C00,003C,0180,F800,0000,037E   ;5834           J7SPEC                          ; ACCORDING TO LAST SPECIFIER
                                        ;5835
                                        ;5836           ;*11----------------------------;
                                        ;5837           SET.V,                          ; SOURCE IS NON ZERO
                                        ;5838           D_Q,                            ; STORE ZERO ALWAYS
U 126B, 0C00,003C,0180,F800,0020,10AE   ;5839           J7CVTOFL                        ;
```

```
                                ;5840   ;Date    : 20-JUN-79                                                    Patch no. 068
                                ;5841   ;Author : R. SHORT
                                ;5842   ;
                                ;5843   ;Source code file  : RPPR
                                ;5844   ;First included in : WCS120
                                ;5845   ;
                                ;5846   ;PROM address to be intercepted : 0CD1
                                ;5847   ;
                                ;5848   ;Description of problem :
                                ;5849   ;       IF A RET IS EXECUTED IN WHICH THE FP POINTS TO A PAGE WHICH IS A
                                ;5850   ;       TRANSLATION BUFFER MISS THE T BIT IS CLEARED
                                ;5851   ;
                                ;5852   ;Instead of that microinstruction, insert here the code which works :
                                ;5853
                                ;5854   118E:   ;-------------------------------;
                                ;5855           LOAD.IB,PC_PC+1,                ; START UP IB TO GET NEXT INST
U 118E, 0810,0038,0180,610C,0000,128A   ;5856   D_RC[T1]                        ; GET THE SAVED PSL FROM D
                                ;5857
                                ;5858           ;-------------------------------;
                                ;5859           R[FP]_Q,LONG,                   ; RESTORE SAVED FP
                                ;5860           ID[PS[]_D,                      ; AND PSL
U 128A, 0001,203C,3D80,3EE8,0000,0062   ;5861   J/IRD                           ; AND GO DO NEXT INST
```

```
                                    ;5862   ;Date   : 02-JUL-79                                                    Patch no. 069
                                    ;5863   ;Author : R. J. Avarbock(30-JUL-80), R. SHORT/T. FOSSUM
                                    ;5864   ;
                                    ;5865   ;Source code file  : DECMAL
                                    ;5866   ;First included in : WCS120
                                    ;5867   ;
                                    ;5868   ;PROM address to be intercepted : 096F
                                    ;5869   ;
                                    ;5870   ;Description of problem :
                                    ;5871   ;       ADDP6   INSTRUCTION
                                    ;5872   ;       THE V BIT IS NOT ALWAYS SET CORRECTLY WHEN THE DEST STRING IS
                                    ;5873   ;       SHORTER THAN THE SOURCE STRINGS
                                    ;5874   ;       S:      .PACKED -6745321
                                    ;5875   ;       S1:     .PACKED +00000000000000000000000000745231
                                    ;5876   ;       D:      .BLKB 15
                                    ;5877   ;               ADDP6 #7,S,#31,S1,#15,D ; SHOULD NOT SET THE V BIT
                                    ;5878   ;                                       ; BEFORE THIS ECO IT WOULD
                                    ;5879   ;       NOTE THAT THIS COULD BE SHARED WITH THE FIX TO ADDP4
                                    ;5880   ;       AT 96D IF PLA TERMS GET REALLY SCARCE
                                    ;5881   ;       Modify to fix v bit problem with negative numbers.
                                    ;5882   ;
                                    ;5883   ;Instead of that microinstruction, insert here the code which works :
                                    ;5884
                                    ;5885   118F:   ;-------------------------------;
                                    ;5886           ALU_0(A),                       ; INITIALISE RC 7 TO ZERO
                                    ;5887           RC[7]_ALU,                      ; TO KEEP TRACK OF OVERFLOW
U 118F, 0003,003C,1980,F9B8,0104,728E ;5888         FE_K[ZERO]                      ; In case no mask gen'ed.   **MOD**
                                    ;5889
                                    ;5890           ;-------------------------------;
                                    ;5891           ALU_D-Q,D_ALU,                  ; DECIMAL ADJUST
                                    ;5892           LAB_R[R4],                      ;
U 128E, 081D,0000,0180,FA20,0000,0BCA ;5893         J/DC.PA_69                      ; RETURN TO PROM
```

```
                                ;5894   ;Date    : 06-JUL-79                                                    Patch no. 070
                                ;5895   ;Author : R. SHORT
                                ;5896   ;
                                ;5897   ;Source code file  : FORKS
                                ;5898   ;First included in : WCS120
                                ;5899   ;
                                ;5900   ;PROM address to be intercepted : 0085
                                ;5901   ;
                                ;5902   ;Description of problem :
                                ;5903   ;       THIS ECO IS BEING ADDED NOW TO ALLOW US TO USE THE SAME PLA
                                ;5904   ;       WHEN THE G AND H FLOATING POINT OPTION IS INSTALLED
                                ;5905   ;       Modified 09-JAN-81 to add 2nd FPLA trap.
                                ;5906   ;
                                ;5907   ;Instead of that microinstruction, insert here the code which works :
                                ;5908
                                ;5909   ;The console examines the least significant bit of the address that 0085 gets
                                ;5910   ;trapped to and uses it to load G&H microcode and to enable/disable the
                                ;5911   ;'G&H present' text that is part of SHOW VERSION console command. A zero in the
                                ;5912   ;lsb of the trap address indicates no G&H a one indicates that G&H is present.
                                ;5913
                                ;5914   ;FPLA trap When there is no G & H
                                ;5915   1180:   ;-------------------------------;
U 1180, 0018,0038,6580,F980,0000,08FC ;5916             RC[T0]_K[.10],J/EXCPT           ; No G&H take fault
                                ;5917
                                ;5918   ;FPLA trap when G & H is present
                                ;5919   11A3:   ;-------------------------------;
                                ;5920           ALU_K[.8000],RC[T4]_ALU,        ; FOR MOVO, SHORT LITERAL DETECT
                                ;5921           CLR.IB.OPC,                     ; CLEAR ESCAPE-CODE,
                                ;5922           PC_PC+1,                        ; INCREMENT PC PAST OP-CODE
U 11A3, C018,0038,4580,F9A4,4000,182A ;5923             J/XFD                           ; AND STROBE INTERRUPTS
```

```
                                        ;5924   ;Date    : 16-JUL-79                                                    Patch no. 071
                                        ;5925   ;Author : R. SHORT
                                        ;5926   ;
                                        ;5927   ;Source code file  : FLOAT
                                        ;5928   ;First included in : WCS120
                                        ;5929   ;
                                        ;5930   ;PROM address to be intercepted : 0471
                                        ;5931   ;
                                        ;5932   ;Description of problem :
                                        ;5933   ;       DIVIDE DOUBLE FLOATING
                                        ;5934   ;       IF THE DIVISOR IS ZERO THE SECOND HALF OF THE RESULT IS NOT
                                        ;5935   ;       STORED AS ZERO
                                        ;5936   ;       DIVD    #0,#^XFFFFFFFFFFFEC98F,FOO
                                        ;5937   ;       SHOULD STORE ^X8000 IN FOO AND 0 IN FOO+4 BEFORE THIS ECO IT WOULD NOT
                                        ;5938   ;
                                        ;5939   ;Instead of that microinstruction, insert here the code which works :
                                        ;5940
                                        ;5941   1190:   ;-------------------------------;
                                        ;5942           R[T1] 0,                        ; CLEAR THE HIGH PART OF RESULT
U 11'', 0003,003E,0180,F988,0000,0010   ;5943           RETURNTO                        ; GO TO STORE RESULT
```

```
                                ;5944   ;Date    : 20-JUL-79                                                    Patch no. 072
                                ;5945   ;Author : T. Fossum
                                ;5946   ;
                                ;5947   ;Source code file  : DECMAL
                                ;5948   ;First included in : WCS120
                                ;5949   ;
                                ;5950   ;PROM address to be intercepted : 0A9F
                                ;5951   ;
                                ;5952   ;Description of problem :
                                ;5953   ;       If the decimal strings contain non-decimal digits, the Subtract-Loop
                                ;5954   ;       in the DIVP-instruction can possibly become an infinite loop, hanging
                                ;5955   ;       up the machine. A case that fails: Divide 0 by the string 1CC814
                                ;5956   ;
                                ;5957   ;Instead of that microinstruction, insert here the code which works :
                                ;5958
                                ;5959   1191:    ;-------------------------------;
                                ;5960           SC_SC+1,                        ; INCREMENT RC-POINTER
                                ;5961           D_D-Q,                          ; DECIMAL ADJUST DIVIDEND
                                ;5962           LAB_R[R15],                     ; LATCH LEADING DIGIT
                                ;5963           Q_0,                            ; FOR USE IN NEXT STATE
U 1191, 081D,0200,01F8,FA78,0080,D265   ;5964           ROR?                            ; TEST FOR BORROW
                                ;5965   =101
                                ;5966           ;101------------------------------; BRANCH ON PSL C-BIT
                                ;5967           R[R15]_LA-K[.1],                ; DECREMENT LEADING DIGIT
U 1265, 0018,8000,0580,FAF8,0070,1267   ;5968           SET.CC(BYTE)                    ;
                                ;5969
                                ;5970           ;111------------------------------;
                                ;5971           ID(SC)_D,                       ; STORE HIGH DIVIDEND LONGWORD
                                ;5972           RC[T5]_ALU,ALU_Q+LC+1,          ; INCREMENT QUOTIENT DIGIT (Q=0)
                                ;5973           CLK.UBCC,BYTE,                  ; SANITY CHECK ON INFINITE LOOPING
                                ;5974           SC_SC+FE,                       ; RESTORE POINTER TO TAIL END OF STRING
U 1267, 0011,A210,0180,35A8,0090,9275   ;5975           ROR?                            ; TEST FOR BORROW FROM LEADING DIGIT
                                ;5976   =101
                                ;5977           ;101------------------------------; BRANCH ON PSL C-BIT
                                ;5978           ALU_Q+LC,RC[T5]_ALU,            ; BORROW, RESTORE QUOTIENT DIGIT
U 1275, 0013,0014,01D0,F9A8,0000,0C36   ;5979           Q_DEC.CON,J/DVAD                ; GENERATE ALL 6'S
                                ;5980
                                ;5981           ;111------------------------------;
                                ;5982           LC_RC(SC),                      ; GET DIVISOR PORTION FOR NEXT SUBTRACT
                                ;5983           ALU_LC,D_ALU,                   ; STORE IT IN D
                                ;5984           Q_ID(SC),                       ; Q GETS DIVIDEND
                                ;5985           EALU_FE,CLK.UBCC,               ; CLOCK RC-COUNT
U 1277, 0810,0138,01F0,2430,0010,70C0   ;5986           Z?                              ; *****POINT OF ECO*****
                                ;5987                                           ; CHECKS IF WE HAVE LOOPED 256 TIMES
                                ;5988   =0
                                ;5989           ;0--------------------------------; BRANCH ON ALU Z-BIT
                                ;5990           ALU_Q[INST.DEP]D,D_ALU,          ; SUBTRACT DIVISOR FROM DIVIDEND
                                ;5991           Q_DEC.CON,                      ; Q GETS DECIMAL CONSTANT TO ADJUST
                                ;5992           SC_SC-K[.1],                    ; DECREMENT ADDRESS POINTER
                                ;5993           LC_RC[T5],                      ; LATCH QUOTIENT DIGIT
                                ;5994           SET.CC(LONG),                   ; CLOCK PSL-C BIT FROM BORROW
U 10C0, 081D,320C,05D0,F928,00F4,AA9B   ;5995           BEN/EALU,J/DVSUB01              ; TEST FOR LAST LONGWORD
                                ;5996
                                ;5997           ;1--------------------------------;NON-DECIMAL DATA INVOLVED,
U 10C1, 0000,003C,0180,F800,0000,0C4C   ;5998           J/DIVP.JUNK.EXIT                ;RESTORE STACK-POINTER AND QUIT
```

```
                                    ;5999   ;Date    : 11-NOV-79                                                    Patch no. 073
                                    ;6000   ;Author : P. R. Guilbault
                                    ;6001   ;
                                    ;6002   ;Source code file  : FORKS
                                    ;6003   ;First included in :
                                    ;6004   ;
                                    ;6005   ;PROM address to be intercepted : 0287
                                    ;6006   ;
                                    ;6007   ;Description of problem :
                                    ;6008   ;       New interlocked Queue instructions. This word used to live in the
                                    ;6009   ;       Q1.MIC file and was moved here so that it would be in the company of
                                    ;6010   ;       all the other words that are trapped by the PLAs. There is no change
                                    ;6011   ;       in operation by this move.
                                    ;6012   ;       TRAPPED AT B-FORK 1st SPECIFIER HAS BEEN EVALUATED, AND IS STORED IN D.
                                    ;6013   ;
                                    ;6014   ;Instead of that microinstruction, insert here the code which works :
                                    ;6015
                                    ;6016   1181:   ;-------------------------------;
                                    ;6017           ALU_D.AND.K[.7],N&Z_ALU.V&C_0,  ; CLOCK LOW 3 ADDRESS BITS TEST FOR
U 1181, 0019,0934,5D80,F800,0050,109A ;6018         IR1?,J/INSERT.Q                 ; FOR INSERT OR REMOVE REJOIN Q1.MIC
```

```
                                ;6019   .TOC    "       PCS microcode patches : Patches first included in WCS122"
                                ;6020
                                ;6021   ;Date    : 05-JUN-80                                                Patch no. 074
                                ;6022   ;Author : T. FOSSUM, P. R. GUILBAULT, R. J. AVARBOCK
                                ;6023   ;
                                ;6024   ;Source code file  : MULP
                                ;6025   ;First included in : WCS122
                                ;6026   ;
                                ;6027   ;PROM address to be intercepted : 0D47
                                ;6028   ;
                                ;6029   ;Description of problem :
                                ;6030   ;       MULP-ROUTINE FAILS WHEN MULTIPLYING LARGE NUMBERS. The problem occurs
                                ;6031   ;       in the MURAW routine which adds the current product just obtained to
                                ;6032   ;       the partial product stored in memory. A carry in, stored in PSL<c>,
                                ;6033   ;       from the previous add is not correctly handled.
                                ;6034   ;
                                ;6035   ;Instead of that microinstruction, insert here the code which works :
                                ;6036
                                ;6037   1192:   ;-------------------------------;
U 1192, 0038,0038,41C0,F800,0000,1291   ;6038           Q_ALU.LEFT,SI/ZERO,ALU_K[.80]   ; Q GETS 100(hex)
                                ;6039
                                ;6040           ;-------------------------------;
                                ;6041           D_D+Q,                          ; ADD PREVIOUS CARRY TO CURRENT PRODUCT
                                ;6042           SET.CC(LONG),                   ; SEE IF THIS PRODUCED A CARRY
U 1291, 081D,0014,01D0,F800,0070,1298   ;6043           Q_DEC.CON                       ; RECORD DECIMAL ADJUSTMENT
                                ;6044
                                ;6045           ;-------------------------------;
                                ;6046           D_D-Q,                          ; DECIMAL ADJUST CURRENT PRODUCT
U 1298, 081D,0000,C9F0,2C00,0000,129A   ;6047           Q_ID[T2]                        ; Q GETS ALL 6'S
                                ;6048
                                ;6049           ;-------------------------------;
                                ;6050           D_D+Q,LAB_R[R15],               ; ADD ALL 6'S TO NEW CURRENT PRODUCT
U 129A, 081D,0214,0180,FA78,0000,1285   ;6051           ROR?                            ; BRANCH ON CARRY FROM ADDING CARRY
                                ;6052   =101
                                ;6053           ;101------------------------------; NO CARRY FROM ADDING CARRY
                                ;6054           D_D+LB,Q_DEC.CON,               ; ADD IN PARTIAL PRODUCT,
U 1285, 080D,0014,01D0,F800,0070,0A3B   ;6055           SET.CC(LONG),J/MURAW            ; CLOCK CARRY FROM THIS.
                                ;6056
                                ;6057           ;111------------------------------; CARRY FROM ADDING CARRY
                                ;6058           D_D+LB,Q_DEC.CON,               ; ADD IN PARTIAL PRODUCT, ALREADY
U 1287, 080D,0014,01D0,F800,0000,0A3B   ;6059           J7MURAW                         ; HAVE CARRY IN PSL<C>
```

```
                                    ;6060   .TOC    "       PCS microcode patches : Patches first included in WCS123"
                                    ;6061
                                    ;6062   ;Date    : 22-MAY-80                                                Patch no. 075
                                    ;6063   ;Author : Paul R. Guilbault
                                    ;6064   ;
                                    ;6065   ;Source code file  : FLOAT
                                    ;6066   ;First included in : WCS123
                                    ;6067   ;
                                    ;6068   ;PROM address to be intercepted : 053F
                                    ;6069   ;
                                    ;6070   ;Description of problem :
                                    ;6071   ;       The warm POLY's do an approximation of 0 whenever the fraction
                                    ;6072   ;       being shifted requires an alignment shift of more than 31(POLYF)
                                    ;6073   ;       or 63(POLYD). This can cuase a rounding error for negative numbers.
                                    ;6074   ;       Note : requires 2 patches
                                    ;6075   ;
                                    ;6076   ;Instead of that microinstruction, insert here the code which works :
                                    ;6077
                                    ;6078                                           ; POLYF
                                    ;6079   115F:   ;-------------------------------; SHIFT > 31. D HAS INCORRECT VALUE
U 115F, 0000,123C,0180,F800,0000,10CE ;6080         EALU?                           ; FIND OUT IF FRAC IS POS OR NEG
                                    ;6081   =1110
                                    ;6082           ;1110---------------------------; SS = 0. POSITIVE FRACTION
                                    ;6083           D_0,                            ; D GETS POSITIVE SIGN
U 10CE, 0F00,003C,0180,F800,0000,0537 ;6084         J7SRCAP0                        ; GO DO ADD
                                    ;6085
                                    ;6086           ;1111---------------------------; SS = 1. NEGATIVE FRACTION
                                    ;6087           D_0-K[.1],                      ; D GETS NEGATIVE SIGN
U 10CF, 081B,0000,0580,F800,0000,0537 ;6088         J7SRCAP0                        ; GO DO ADD
```

```
                                        ;6089   ;Date    : 29-APR-80                                                 Patch no. 076
                                        ;6090   ;Author : Paul R. Guilbault
                                        ;6091   ;
                                        ;6092   ;Source code file  : FLOAT
                                        ;6093   ;First included in : WCS123
                                        ;6094   ;
                                        ;6095   ;PROM address to be intercepted : 055F
                                        ;6096   ;
                                        ;6097   ;Description of problem :
                                        ;6098   ;       Continuation of POLY rounding problem
                                        ;6099   ;
                                        ;6100   ;Instead of that microinstruction, insert here the code which works :
                                        ;6101
                                        ;6102                                           ; POLYD
                                        ;6103   1193:   ;-------------------------------; 63 < EXP DIFF
                                        ;6104           D_Q,                            ; D GETS SIGN, SC GETS BIGGER EXPONENT
                                        ;6105           SC_LC(EXP),                     ; FAKEM OUT BY PUTTING SIGN IN BOTH
U 1193, 0C10,003A,0180,F800,0083,0021   ;6106           RETURN[21]                      ; HALVES AND RETURNING AS IF
                                        ;6107                                           ; 32 <= EXP DIFF <= 64
                                        ;6108
                                        ;6109   ; This will bypasses the optimization for very samll fractions. Although the
                                        ;6110   ; optimization is still valid whenever "COEF > PROD" or when
                                        ;6111   ; "(COEF < PROD).and.(COEF is positive)", it is easier to always do the add/sub.
```

```
                                ;6112   ;Date   : 12-Jun-80                                                  Patch no. 077
                                ;6113   ;Author : Robert J. Avarbock
                                ;6114   ;
                                ;6115   ;Source code file  : DECMAL
                                ;6116   ;First included in : WCS123
                                ;6117   ;
                                ;6118   ;PROM address to be intercepted : 01B5
                                ;6119   ;
                                ;6120   ;Description of problem :
                                ;6121   ;       ASHP doesn't set condition codes properly when it runs out
                                ;6122   ;       of destination string when it has yet to see non-zero source
                                ;6123   ;       data, whether actual source data or shifted in zeros.
                                ;6124   ;       This also causes the source sign to get lost.
                                ;6125   ;
                                ;6126   ;Instead of that microinstruction, insert here the code which works :
                                ;6127
                                ;6128   1195:   ;-----------------------------;
                                ;6129           ALU_RC[T2],CLK.UBCC,            ; Get RC2 in case no OV yet...
U 1195, 0010,1638,0180,F810,0010,10DE   ;6130           STATE7-4?                       ; but don't bother if neg shf cnt.
                                ;6131
                                ;6132   =1110   ;1110-----------------------------; POS SHF CNT - Try it.
U 10DE, 0000,1B3C,0180,F800,0000,10EB   ;6133           ALU?,J/RJA.ASHP.ECO0            ; Now, go branch on ALU<Z>
                                ;6134
                                ;6135           ;1111-----------------------------; NEG SHF CNT - No overflow.
                                ;6136           ALU_0(A),R[R0]_ALU,             ; Clear R0  (This Uword is ASH.F2)
                                ;6137           N_AMX.Z_TST,                    ; Clear N bit.
                                ;6138           STATE_STATE.ANDNOT.K[.30],      ; Use these bits in finish-routine.
U 10DF, 0003,173C,7980,FA80,1434,488D   ;6139           STATE3-0?,J/FINI1               ; test sign-bit.
                                ;6140
                                ;6141   =1011
                                ;6142   RJA.ASHP.ECO0:
                                ;6143           ;1011-----------------------------;
U 10EB, 0000,003C,3180,F800,1404,30EF   ;6144           STATE_STATE.OR.K[.40]           ; Non-Zero RC2 - set overflow.
                                ;6145
                                ;6146           ;1111-----------------------------; Nothing in RC2 - no overflow.
                                ;6147           ALU_0(A),R[R0]_ALU,             ; Clear R0  (This Uword is ASH.F2)
                                ;6148           N_AMX.Z_TST,                    ; Clear N bit.
                                ;6149           STATE_STATE.ANDNOT.K[.30],      ; Use these bits in finish-routine.
U 10EF, 0003,173C,7980,FA80,1434,488D   ;6150           STATE3-0?,J/FINI1               ; test sign-bit.
```

```
                                    ;6151   ;Date    : 02-JUL-80                                                    Patch no. 078
                                    ;6152   ;Author : K. CASSIDY
                                    ;6153   ;
                                    ;6154   ;Source code file  : MTPR
                                    ;6155   ;First included in :
                                    ;6156   ;
                                    ;6157   ;PROM address to be intercepted : 0D61
                                    ;6158   ;
                                    ;6159   ;Description of problem :
                                    ;6160   ;       WHEN LOADING THE SCBB USING THE MTPR INSTRUCTION,
                                    ;6161   ;       THE PRESENT MICROCODE CHECKS BITS  26-27 OF THE NUMBER BEING LOADED
                                    ;6162   ;       CAUSING A RESERVED OPERAND FAULT IF THEY ARE NOT BOTH ZERO.
                                    ;6163   ;       THIS IMPROPERLY RESTRICTS THE PLACEMENT OF THE CONTROL BLOCK,
                                    ;6164   ;       AS SPECIFICATIONS STATE BITS 30-31 ARE TO BE CHECKED.
                                    ;6165   ;       THIS PATCH CAUSES THE CORRECT CHECK TO TAKE PLACE.
                                    ;6166   ;
                                    ;6167   ;Instead of that microinstruction, insert here the code which works :
                                    ;6168
                                    ;6169   117E:   ;-------------------------------;
U 117E, 0000,003C,8580,F800,0084,929C ;6170         SC_SC+K[.C]                     ;SC=REG# +C
                                    ;6171
                                    ;6172         ;-------------------------------;
                                    ;6173         ALU_D.AND.K[.C0],               ;CHECK THAT BITS 31-30 ARE ZERO
                                    ;6174         CLK.UBCC,                       ;WITH ALU Z
                                    ;6175         D_D.SWAP,                       ;RESTORE NORMAL BIT ORDER
U 129C, 0B19,0134,D180,F800,0010,08C0 ;6176         Z?,J/MO.PA.78                   ;ARE BITS 1-0 ZERO?
```

```
                                ;6177   ;Date    : 09-Jul-80                                                    Patch no. 079
                                ;6178   ;Author : Robert J. Avarbock
                                ;6179   ;
                                ;6180   ;Source code file  : DECMAL
                                ;6181   ;First included in : WCS123
                                ;6182   ;
                                ;6183   ;PROM address to be intercepted : 0BD3
                                ;6184   ;
                                ;6185   ;Description of problem :
                                ;6186   ;       ADDP4/6 SUBP4/6 sometime return -0 when they should
                                ;6187   ;       return +0.  When they try to change the sign, if the actual
                                ;6188   ;       write to memory fails because of memory management, when the
                                ;6189   ;       instruction starts up again, it sees that the 1 time bit has
                                ;6190   ;       been cleared which causes the instruction to think it is
                                ;6191   ;       the first time through.  It tries to do things that eventually
                                ;6192   ;       lead it to not actually changing the sign.
                                ;6193   ;
                                ;6194   ;Instead of that microinstruction, insert here the code which works :
                                ;6195
                                ;6196   1196:   ;-------------------------------;
                                ;6197           ALU_LA-0-1,VAK/LOAD,            ;  Get address of sign byte.
                                ;6198           STATE_STATE.OR.K[.4],           ;  Set 1 time bit so if the write
U 1196, 001C,0008,1180,F800,1604,2BD4   ;6199           J/DC.PA.79                      ;  to the sign bit faults, it will
                                ;6200                                           ;  get done correctly next time thru.
```

```
                                ;6201   ;Date    : 30-JUN-80                                                    Patch no. 080
                                ;6202   ;Author : P. R. Guilbault
                                ;6203   ;
                                ;6204   ;Source code file  : FLOAT
                                ;6205   ;First included in :
                                ;6206   ;
                                ;6207   ;PROM address to be intercepted : 014C, 014E, 017E
                                ;6208   ;
                                ;6209   ;Description of problem :
                                ;6210   ;       Convert from floating arithmetic traps to floating arithmetic faults.
                                ;6211
                                ;6212   ;(Formally EXPCKR) - WARM FLOATING OVERFLOW/UNDERFLOW EXPONENT CHECK FOR :
                                ;6213   ;       ADDF2/SUBF2     Short literal, Register
                                ;6214   ;                       Register,      Register
                                ;6215   ;       ADDF3/SUBF3     *****,         Short literal, Register
                                ;6216   ;                       *****,         Register,      Register
                                ;6217   ;Result stored, register no. of dest is on RLOG, original dest is in RCTO
                                ;6218
                                ;6219   1147:   ;-------------------------------;
                                ;6220           EALU_SC,R(SP1)_PACK.FP,         ;PACK RESULT
                                ;6221           SET.CC(INST),                   ;SET COND CODES
U 1147, 4008,D438,0180,F8C5,4070,10F1   ;6222           CLR.IB0-1,PC_PC+2,SC?           ;UPDATE PC, POP IB, CK IF UNDER/OVFL
                                ;6223   =0000
                                ;6224   =0001   ;0001-------------------------; SC.eq.0, UNDERFLOW
U 10F1, 0000,003D,3DF0,2C00,0000,12A1   ;6225           Q_ID[PSL],CALL[FIXUP]           ; GET PSL TO CHECK PSL<fu>
                                ;6226
                                ;6227   =0011   ;0011-------------------------; SC.eq.[01 to FF], NO OVER/UNDERFLOW
U 10F3, F80C,003B,01F1,F857,139B,6000   ;6228           IRD                             ;
                                ;6229
                                ;6230   =0101   ;0101-------------------------; SC.lss.0, UNDERFLOW
U 10F5, 0000,003D,3DF0,2C00,0000,12A1   ;6231           Q_ID[PSL],CALL[FIXUP]           ; GET PSL TO CHECK PSL<fu>
                                ;6232
                                ;6233   =0111   ;0111-------------------------; SC.gt.FF, OVERFLOW
U 10F7, 0018,0039,0180,F9B8,0000,12A1   ;6234           RC[T7]_K[.8],CALL[FIXUP]        ; T7 GETS OVERFLOW TRAP CODE
                                ;6235
                                ;6236   =1101   ;1101-------------------------; RETURN HERE FROM FIXUP UNDERFLOW
U 10FD, 0001,203C,0180,F800,1408,729E   ;6237           STATE_Q(EXP),J/CHK11            ; STATE<0> GETS PSL<fu>
                                ;6238
                                ;6239   CHK10:  ;1110-------------------------; UNDERFLOW AND PSL<fu>=0
                                ;6240           R(SC)_K[ZERO],N&Z_ALU.V&C_0,    ; DEST GETS 0, PRETEND IT DIDN'T HAPPEN
U 10FE, 0018,0038,1980,F8E8,0050,0062   ;6241           J/IRD                           ;
                                ;6242
                                ;6243           ;1111-------------------------; RETURN HERE FROM FIXUP OVERFLOW
                                ;6244           R(SC)_D,                        ; OR HERE FOR UNDERFLOW WITH PSL<fu>=1
U 10FF, 0001,003C,0180,F8E8,0000,12B4   ;6245           J/FLOAT.FAULT                   ; RESTORE DESTINATION OPERAND
                                ;6246
                                ;6247   CHK11:  ;-------------------------------; T7 GETS UNDERFLOW TRAP CODE
U 129E, 0018,1738,F580,F9B8,0000,10FE   ;6248           RC[T7]_K[.A],STATE0?,J/CHK10    ; PSL<fu> SET?
                                ;6249
                                ;6250   ;SUBROUTINE TO RESTORE REG # FROM RLOG AND TO GET SAVED DEST FROM TO INTO D
                                ;6251   FIXUP:  ;-------------------------------; GET REG # INTO D<10:07> AND
U 12A1, 0840,0038,01A8,F800,1C00,12A4   ;6252           D_RLOG.RIGHT,Q_Q.LEFT           ; GET PSL<fu> INTO Q<7>
                                ;6253
                                ;6254           ;-------------------------------;
U 12A4, 0811,003A,0180,F900,0088,600C   ;6255           SC_D(EXP)(A),D_RC[TO],RETURN[0C]; PUT REG # IN SC AND ORIG DEST IN D
```

```
                                ;6256   ;Date   : 30-JUN-80                                                    Patch no. 081
                                ;6257   ;Author : P. R. Guilbault
                                ;6258   ;
                                ;6259   ;Source code file  : FLOAT
                                ;6260   ;First included in :
                                ;6261   ;
                                ;6262   ;PROM address to be intercepted : 013E
                                ;6263   ;
                                ;6264   ;Description of problem :
                                ;6265   ;       Convert from floating arithmetic traps to floating arithmetic faults.
                                ;6266
                                ;6267   ;(Formally EXPCKP) - WARM FLOATING OVERFLOW/UNDERFLOW EXPONENT CHECK FOR :
                                ;6268   ;       ADDF2/SUBF2     *****, Register
                                ;6269   ;       MULF2/DIVF2     *****, Register
                                ;6270   ;Result already stored in R(PRN)
                                ;6271
                                ;6272   114F:   ;-------------------------------;
                                ;6273           EALU_SC,R(PRN)_PACK.FP,         ; PACK RESULT
                                ;6274           SET.CC(INST),                   ; SET COND CODES
U 114F, C008,D438,0180,F8DC,4070,1101   ;6275           CLR.IB.OPC,PC_PC+1,SC?          ; UPDATE PC, POP IB, CK IF UNDER/OVFL
                                ;6276
                                ;6277   =0001   ;0001----------------------------; SC.eq.0, UNDERFLOW
U 1101, 0810,0039,0180,F900,0000,12AC   ;6278           D_RC[TO],CALL[PSLFU]            ; D GETS ORIGINAL DEST
                                ;6279
                                ;6280           ;0011----------------------------; SC.eq.[01 to FF], NO OVER/UNDERFLOW
U 1103, F80C,0038,01F1,F857,139B,6000   ;6281           IRD                             ; 01 TO FF FOR EXP, OK
                                ;6282
                                ;6283           ;0101----------------------------; SC.lss.0, UNDERFLOW
U 1105, 0810,0039,0180,F900,0000,12AC   ;6284           D_RC[TO],CALL[PSLFU]            ; D GETS ORIGINAL DEST
                                ;6285
                                ;6286           ;0111----------------------------; SC.gt.FF, OVERFLOW
U 1107, 0810,0038,0180,F900,0000,12A6   ;6287           D_RC[TO],J/CHK20                ; D GETS ORIGINAL DEST
                                ;6288
                                ;6289   =1101   ;1101----------------------------; RETURN HERE IF PSL<fu>.eq.0(SC.eq.0)
                                ;6290           R(PRN)_K[ZERO],N&Z_ALU.V&C_0,   ; PRETEND UNDERFLOW DIDN'T HAPPEN
U 110D, 0018,0038,1980,F8D8,0050,0062   ;6291           J/IRD                           ;
                                ;6292
                                ;6293           ;1111----------------------------; RETURN HERE IF PSL<fu>.eq.1(SC.ne.0)
U 110F, 0018,0038,F580,F9B8,0000,12A8   ;6294           RC[T7]_K[.A],J/CHK21            ; T7 GETS UNDERFLOW TRAP CODE
                                ;6295
                                ;6296   CHK20:  ;-------------------------------;
U 12A6, 0018,0038,0180,F9B8,0000,12A8   ;6297           RC[T7]_K[.8]                    ; T7 GETS OVERFLOW TRAP CODE
                                ;6298
                                ;6299   CHK21:  ;-------------------------------;
U 12A8, 0001,003C,0180,F8D8,0000,12B4   ;6300           R(PRN)_D,J/FLOAT.FAULT          ; RESTORE ORIGINAL DEST BEFORE FAULT
                                ;6301
                                ;6302   ;Subroutine to isolate PSL<fu>
                                ;6303   PSLFU:  ;-------------------------------;
U 12AC, 0000,003C,3DF0,2C00,0000,12AE   ;6304           Q_ID[PSL]                       ; GET PSL
                                ;6305
                                ;6306   PSLFU.A:;-------------------------------;
U 12AE, 0019,2034,3180,F800,0082,12B3   ;6307           SC_Q.AND.K[.40]                 ; ISOLATE PSL<fu> IN SC
                                ;6308
                                ;6309           ;----------------------------; CONDITIONAL RETURN ON
U 12B3, 0000,123E,0180,F800,0000,000D   ;6310           EALU?,RETURN[0D]                ; SC.NE.0(PSL<fu>)
```

```
                                ;6311   ;Date   : 30-JUN-80                                                    Patch no. 082
                                ;6312   ;Author : P. R. Guilbault
                                ;6313   ;
                                ;6314   ;Source code file  : FLOAT
                                ;6315   ;First included in :
                                ;6316   ;
                                ;6317   ;PROM address to be intercepted : 01DC, 01DE, 043C
                                ;6318   ;
                                ;6319   ;Description of problem :
                                ;6320   ;       Convert from floating arithmetic traps to floating arithmetic faults.
                                ;6321
                                ;6322   ;(Formally EXPCK) - WARM FLOATING OVERFLOW/UNDERFLOW EXPONENT CHECK FOR :
                                ;6323   ;
                                ;6324   ;       ADDF2/SUBF2     Memory destination.
                                ;6325   ;
                                ;6326   ;       ADDF3/SUBF3     Register destination(except LIT to REG or REG to REG)
                                ;6327   ;
                                ;6328   ;       ADDF3/SUBF3     Memory destination.
                                ;6329   ;
                                ;6330   ;       MULF2/DIVF2     Memory destination.
                                ;6331   ;
                                ;6332   ;       MULF3/DIVF3
                                ;6333   ;
                                ;6334   ;       CVTDF
                                ;6335   ;
                                ;6336   ;Result in D, destination not evaluated yet
                                ;6337
                                ;6338   1150:   ;-------------------------------;
                                ;6339           EALU_SC,D_PACK.FP,              ;ADDF2/SUBF2: PACK RESULT
                                ;6340           SET.CC(INST),                   ;
U 1150, 0808,D438,0180,F800,0070,1131   ;6341   SC?                             ;CK IF UNDERFL OR OVFL
                                ;6342
                                ;6343   =0001   ;0001-----------------------------; SC.eq.0, UNDERFLOW
                                ;6344           D_K[ZERO],N&Z_ALU.V&C_0,        ; SET CC'S & RESULT=0 IN CASE PSF<fu>=0
U 1131, 0818,0039,1980,F800,0050,12AC   ;6345   CALL[PSLFU]                      ;
                                ;6346
                                ;6347           ;0011-----------------------------; SC.eq.[01 to FF], NO OVER/UNDERFLOW
U 1133, F000,003F,01F0,F847,0000,0300   ;6348   WRITE.DEST                       ; NO OVER/UNDERFLOW GO WRITE DEST
                                ;6349
                                ;6350           ;0101-----------------------------; SC.lss.0, UNDERFLOW
                                ;6351           D_K[ZERO],N&Z_ALU.V&C_0,        ; SET CC'S & RESULT=0 IN CASE PSF<fu>=0
U 1135, 0818,0039,1980,F800,0050,12AC   ;6352   CALL[PSLFU]                      ;
                                ;6353
                                ;6354           ;0111-----------------------------; SC.gt.FF, OVERFLOW
U 1137, 0018,0038,0180,F988,0000,1284   ;6355   RC[T7]_K[.8],J/FLOAT.FAULT        ; T7 GETS OVERFLOW TRAP CODE
                                ;6356
                                ;6357   =1101   ;1101-----------------------------; RETURN HERE IF PSL<fu>.eq.0(SC.eq.0)
U 113D, F000,003F,01F0,F847,0000,0300   ;6358   WRITE.DEST                       ; PRETEND UNDERFLOW DIDN'T HAPPEN
                                ;6359
                                ;6360           ;1111-----------------------------; RETURN HERE IF PSL<fu>.eq.1(SC.ne.0)
U 113F, 0018,0038,F580,F988,0000,1284   ;6361   RC[T7]_K[.A],J/FLOAT.FAULT        ; T7 GETS UNDERFLOW TRAP CODE
```

```
                                ;6362   ;Date    : 30-JUN-80                                                    Patch no. 083
                                ;6363   ;Author : P. R. Guilbault
                                ;6364   ;
                                ;6365   ;Source code file  : FLOAT
                                ;6366   ;First included in :
                                ;6367   ;
                                ;6368   ;PROM address to be intercepted : 03E4
                                ;6369   ;
                                ;6370   ;Description of problem :
                                ;6371   ;       Convert from floating arithmetic traps to floating arithmetic faults.
                                ;6372
                                ;6373   ;(Formally PACKD.4) - WARM FLOATING OVERFLOW/UNDERFLOW EXPONENT CHECK FOR :
                                ;6374   ;
                                ;6375   ;       ADDD2, ADDD3, SUBD2, SUBD3, MULD2, MULD3, DIVD2, DIVD3
                                ;6376
                                ;6377   117F:   ;-------------------------------;
                                ;6378           SC_K[.10].ALU,                  ; SC GETS 16. FOR WORD SWAP FOR FRAC <L>
                                ;6379           F=_SC,                          ; SAVE EXP FOR SETTING COND CODES LATTER
                                ;6380           Q_D,                            ; READY FOR SWAP WORD FOR FRAC <L>
U 117F, 0018,1438,65E0,F800,0182,1201   ;6381   SC?                             ; EXP SHOWS 0, NON-0, UNDRFLOW, OVRFLOW
                                ;6382
                                ;6383   =0001   ;0001----------------------------; SC.eq.0, UNDERFLOW
                                ;6384           D_0,Q_0,EALU_K[ZERO],           ; SET RESULT <H,L> TO ZEROS
                                ;6385           RC[T1]_K[ZERO],                 ; RESULT <L> SET TO 0
                                ;6386           SET.CC(INST),                   ; SET COND CODES
U 1201, 0F18,C039,19F8,F988,0074,72AC   ;6387   CALL[PSLFU]                      ; GO SEE IF PSL<fu> IS SET
                                ;6388
                                ;6389           ;0011----------------------------; SC.eq.[01 to FF], NO OVER/UNDERFLOW
                                ;6390           D_DAL.SC,                       ; SWAP WORD
U 1203, 0D10,0038,01C0,F908,0000,0771   ;6391   Q_RC[T1],J/PACKD.6               ; Q GETS RESULT <H>
                                ;6392
                                ;6393           ;0101----------------------------; SC.lss.0, UNDERFLOW
                                ;6394           D_0,Q_0,EALU_K[ZERO],           ; SET RESULT <H,L> TO ZEROS
                                ;6395           RC[T1]_K[ZERO],                 ; RESULT <L> SET TO 0
                                ;6396           SET.CC(INST),                   ; SET COND CODES
U 1205, 0F18,C039,19F8,F988,0074,72AC   ;6397   CALL[PSLFU]                      ; GO SEE IF PSL<fu> IS SET
                                ;6398
                                ;6399           ;0111----------------------------; SC.gt.FF, OVERFLOW
U 1207, 0018,0038,0180,F9B8,0000,12B4   ;6400   RC[T7]_K[.8],J/FLOAT.FAULT        ; T7 GETS OVERFLOW TRAP CODE
                                ;6401
                                ;6402   =1101   ;1101----------------------------; RETURN HERE IF PSL<fu>.eq.0(SC.eq.0)
U 120D, 0000,003E,0180,F800,0000,0010   ;6403   RETURN10                         ; PRETEND UNDERFLOW DIDN'T HAPPEN
                                ;6404
                                ;6405           ;1111----------------------------; RETURN HERE IF PSL<fu>.eq.1(SC.ne.0)
U 120F, 0018,0038,F580,F9B8,0000,12B4   ;6406   RC[T7]_K[.A],J/FLOAT.FAULT        ; T7 GETS UNDERFLOW TRAP CODE
```

```
                                    ;6407   ;Date    : 08-JUL-80                                                    Patch no. 084
                                    ;6408   ;Author : P. R. Guilbault
                                    ;6409   ;
                                    ;6410   ;Source code file  : FLOAT
                                    ;6411   ;First included in :
                                    ;6412   ;
                                    ;6413   ;PROM address to be intercepted : 05BE
                                    ;6414   ;
                                    ;6415   ;Description of problem :
                                    ;6416   ;       Convert from floating arithmetic traps to floating arithmetic faults.
                                    ;6417
                                    ;6418   ;(Formally ACBF.4) - WARM FLOATING OVERFLOW/UNDERFLOW EXPONENT CHECK FOR : ACBF
                                    ;6419
                                    ;6420   1$94:    ;-------------------------------;NORMAL COMPLETION OF FLOATING ADD
                                    ;6421           EALU_SC,D_PACK.FP,              ;REBUILD FLOATING RESULT
                                    ;6422           SET.CC(INST),                   ;SET CONDITION CODES ON IT
U 1194, 0808,D438,0180,F800,0070,1211 ;6423         SC?                             ;GO TEST FOR OVER/UNDERFLOW
                                    ;6424
                                    ;6425   =0001   ;0001--------------------------; SC.eq.0, UNDERFLOW
                                    ;6426           D_K[ZERO],SET.CC(INST),         ; RETURN ZERO ON UNDERFLOW
U 1211, 0818,C039,1980,F800,00F4,72AC ;6427         SC_K[ZERO],CALL[PSLFU]          ; CLEAR OVERFLOW FLAG
                                    ;6428
                                    ;6429           ;0011--------------------------; SC.eq.[01 to FF], NO OVER/UNDERFLOW
                                    ;6430           ALU_D.XOR.R[R15],SS_ALU15,      ; DIFF OF ADDEND & INDEX SIGNS TO SS
                                    ;6431           SC_K[ZERO],                     ; CLEAR OVERFLOW FLAG
U 1213, 000D,1720,1981,FA78,0084,6598 ;6432         STATE0?,J/ACBF.5                ; WHERE IS RESULT STORED?
                                    ;6433
                                    ;6434           ;0101--------------------------; SC.lss.0, UNDERFLOW
                                    ;6435           D_K[ZERO],SET.CC(INST),         ; RETURN ZERO ON UNDERFLOW
U 1215, 0818,C039,1980,F800,00F4,72AC ;6436         SC_K[ZERO],CALL[PSLFU]          ; CLEAR OVERFLOW FLAG
                                    ;6437
                                    ;6438           ;0111--------------------------; SC.gt.FF, OVERFLOW
U 1217, 0018,0038,0180,F988,0000,12B4 ;6439         RC[T7]_K[.8],J/FLOAT.FAULT      ; T7 GETS OVERFLOW TRAP CODE
                                    ;6440
                                    ;6441   =1101   ;1101--------------------------; RETURN HERE IF PSL<fu>.eq.0(SC.eq.0)
                                    ;6442           ALU_R[R15],SS_ALU15,            ; PRETEND UNDERFLOW DIDN'T HAPPEN
U 121D, 0000,173C,0181,FA78,0000,0598 ;6443         STATE0?,J/ACBF.5                ; ADDEND SIGN TO SS
                                    ;6444
                                    ;6445           ;1111--------------------------; RETURN HERE IF PSL<fu>.eq.1(SC.ne.0)
U 121F, 0018,0038,F580,F988,0000,12B4 ;6446         RC[T7]_K[.A],J/FLOAT.FAULT      ; T7 GETS UNDERFLOW TRAP CODE
```

```
                                ;6447   ;Date    : 30-JUN-80                                                    Patch no. 085
                                ;6448   ;Author : P. R. Guilbault
                                ;6449   ;
                                ;6450   ;Source code file  : FLOAT
                                ;6451   ;First included in :
                                ;6452   ;
                                ;6453   ;PROM address to be intercepted : 0E00
                                ;6454   ;
                                ;6455   ;Description of problem :
                                ;6456   ;       Convert from floating arithmetic traps to floating arithmetic faults.
                                ;6457
                                ;6458   ;(Formally DIVBY0) - WARM FLOATING POINT DIVIDE BY ZERO CHECK FOR :
                                ;6459   ;
                                ;6460   ;       DIVF2, DIVF3, DIVD2, DIVD3
                                ;6461
                                ;6462   1151:   ;-------------------------------;
U 1151, 0018,0038,D980,F9B8,0000,12B4 ;6463           RC[T7]_K[.9],J/FLOAT.FAULT      ; T7 GETS DIVIDE BY 0 TRAP CODE
                                ;6464
                                ;6465
                                ;6466
                                ;6467   ;Enter here with floating fault code in T7.
                                ;6468   FLOAT.FAULT:
                                ;6469           ;-------------------------------;
                                ;6470           RC[T0]_K[.34],                  ; T0 VECTOR ID OF FLOATING FAULTS
U 12B4, 001B,0038,2980,F980,1408,72A0 ;6471     STATE_0(A)                      ; CLEAR STATE(WILL BE SET TO 1 LATER)
                                ;6472
                                ;6473   =00    ;00-------------------------------;
                                ;6474           RC[PC.SV]_PC,                   ; SET UP PC WHERE 'BAKUP.PC'' WANTS IT
U 12A0, 0014,1539,0180,F9E0,0000,0DA7 ;6475     RLOG.EMPTY?,CALL[BAKUP.RGS]     ; GO BACK UP REGS & PC
                                ;6476
                                ;6477   =10    ;10-------------------------------; RETURN HERE FROM 'BAKUP.RGS''
                                ;6478           Q_ID[PSL],                      ; GET PSL INTO Q
                                ;6479           STATE_STATE+1,                  ; STATE=1 TO INDICATE PARAMETERS
U 12A2, 0000,003D,3DF0,2C00,1400,CDE8 ;6480     CALL[EXCPT1]                    ; GO INITIALIZE FAULT
                                ;6481
                                ;6482           ;11-------------------------------; RETURN HERE FROM 'EXCPT1''
U 12A3, 0810,0038,0180,F938,0000,13F4 ;6483     D_RC[T7],J/POLY.FLOAT.FAULT.A   ; GET FAULT CODE
```

```
                                ;6484   ;Date    : 22-JUL-80                                                    Patch no. 086
                                ;6485   ;Author : K. CASSIDY
                                ;6486   ;
                                ;6487   ;Source code file  : EDITPC
                                ;6488   ;First included in :
                                ;6489   ;
                                ;6490   ;PROM address to be intercepted : 0A62
                                ;6491   ;
                                ;6492   ;Description of problem :
                                ;6493   ;       THE EDIT PC INSTRUCTION INCORRECTLY CHANGES THE CONTENTS
                                ;6494   ;       OF THE STATE REGISTER FROM 2 TO 3 WHEN RETURNING FROM AN INTERRUPT
                                ;6495   ;       VIA STATE 2. THE STATE SHOULD BE RESET TO 1. THIS CORRECTION ALLOWS
                                ;6496   ;       THIS TO HAPPEN.
                                ;6497   ;
                                ;6498   ;Instead of that microinstruction, insert here the code which works :
                                ;6499
                                ;6500   119F:   ;--------------------------------;
                                ;6501           VA_LA-K[.1],                    ;DECREMENT THE PATTERN ADDRESS
                                ;6502           R[R3]_LA-K[.1],                 ;IN R3 AND VA
                                ;6503           Q_0,                            ;
                                ;6504           STATE_K[.1],                    ;SET STATE TO 1
                                ;6505           INTRPT.STROBE,                  ;
U 119F, 0018,0000,05F8,FA98,5604,6AB6   ;6506           J/EDPATTIRST                    ;
```

```
                                ;6507   ;Date    : 25-JUL-80                                                    Patch no. 087
                                ;6508   ;Author :  PAUL GUILBAULT, K. CASSIDY
                                ;6509   ;
                                ;6510   ;Source code file  : FLOAT
                                ;6511   ;First included in :
                                ;6512   ;
                                ;6513   ;PROM address to be intercepted : 067E
                                ;6514   ;
                                ;6515   ;Description of problem :
                                ;6516   ;       THE DESTINATION REGISTER IS NOT BEING RESTORED TO ITS ORIGINAL VALUE
                                ;6517   ;       WHEN A FAULT OCCURS DURING A DIVF2 INSTRUCTION WITH REGISTER
                                ;6518   ;       REFERENCE MODE FOR THE DESTINATION. THIS PATCH STORES Q IN [T1]
                                ;6519   ;       INSTEAD OF [T0] SO [T0] CAN BE USED TO STORE THE ORIGINAL CONTENTS
                                ;6520   ;       OF THE DESTINATION SO IT CAN BE RESTORED BY LATER CODE IF REQUIRED.
                                ;6521   ;       THIS PATCH ALSO INCORPORATES PATCH NO. 022.
                                ;6522   ;
                                ;6523   ;Instead of that microinstruction, insert here the code which works :
                                ;6524
                                ;6525   1197:   ;-------------------------------;
U 1197, 0001,323C,0180,F988,0181,1119   ;6526           SC_FE,FE_SC,RC[T1]_Q,EALU?      ;SWAP EXPS
                                ;6527
                                ;6528   =1001   ;1001----------------------------;
U 1119, 0000,003C,0180,F800,0800,03D0   ;6529           ALU_LA,CHK.FLT.OPR,J/DIVF6      ;D'SOR = 0:  NO DIVIDE
                                ;6530
                                ;6531           ;1011---------- ----------------;
                                ;6532           FE_SC-FE,Q_0,RC[T0]_LA,         ;D'SOR TO LB
U 111B, 0000,003C,01F8,F980,0100,B0C4   ;6533           J/PA.DIVF3                      ;
                                ;6534
                                ;6535           ;1101----------------------------;
U 111D, 0000,003C,0180,F800,0800,03D0   ;6536           ALU_LA,CHK.FLT.OPR,J/DIVF6      ;D'SOR = 0:  NO DIVIDE
                                ;6537
                                ;6538           ;1111----------------------------;
U 111F, 0000,003C,0180,F800,0800,0069   ;6539           ALU_LA,CHK.FLT.OPR,J/MULF.0     ;D'END = 0
                                ;6540   =0
                                ;6541   PA.DIVF3:
                                ;6542           ;0-------------------------------;
                                ;6543           D_LA(FRAC),                     ;GET D'END FRAC, RESULT SIGN
                                ;6544           SS_SS.XOR.ALU15&SD_ALU15,       ;
                                ;6545           LC_RC[T1],CHK.FLT.OPR,          ;D'SOR TO LB, SET LOOP CT FOR 25.
U 10C4, 0900,003D,B985,F908,0884,6327   ;6546           SC_K[.19],CALL,J/DIVFX          ;
                                ;6547
                                ;6548           ;1--------------------------------;
U 10C5, 0000,1B3C,0180,F800,0000,013E   ;6549           ALU?,J/ADDFDX                   ;TEST FOR NORM, PACK AND STORE
```

```
                                    ;6550   ;Date   : 25-JUL-80                                                    Patch no. 088
                                    ;6551   ;Author :  PAUL GUILBAULT, K. CASSIDY
                                    ;6552   ;
                                    ;6553   ;Source code file  : FLOAT
                                    ;6554   ;First included in :
                                    ;6555   ;
                                    ;6556   ;PROM address to be intercepted : 006B
                                    ;6557   ;
                                    ;6558   ;Description of problem :
                                    ;6559   ;       THE DESTINATION REGISTER IS NOT BEING RESTORED TO ITS ORIGINAL VALUE
                                    ;6560   ;       WHEN A FAULT OCCURS DURING A MULF2 INSTRUCTION WITH REGISTER
                                    ;6561   ;       REFERENCE MODE FOR THE DESTINATION. THIS PATCH STORES THE INITAL
                                    ;6562   ;       VALUE OF THE DESTINATION IN RC[TO] SO THAT LATER CODE CAN RESTORE
                                    ;6563   ;       IT IF REQUIRED.NOTE THAT THE SUBROUTINE MULFX IS REPRODUCED IN
                                    ;6564   ;       THE PATCH.THIS WAS DONE TO AVOID TRAPPING AND THUS SLOWING
                                    ;6565   ;       DOWN OTHER MULTIPLIES THAT USE MULFX.WHEN PATCH IS INCORPORATED
                                    ;6566   ;       INTO PCS ONLY THE CHANGES LABLED ARE NEEDED.
                                    ;6567   ;
                                    ;6568   ;Instead of that microinstruction, insert here the code which works :
                                    ;6569
                                    ;6570   1198:   ;01011-------------------------;
                                    ;6571           D_LA(FRAC),                     ;GET DST FRAC
                                    ;6572           LC_RC[TO],Q_0,                  ;
                                    ;6573           SC_K[.FFF9],                    ;GET SHIFT VALUE -7
                                    ;6574           SS_SS.XOR.ALU15&SD_ALU15,       ;GET RESULTANT SIGN TO SS
U 1198, 0900,003C,BDFD,F90C,0084,72BA ;6575         J/PA.MULFX                      ;
                                    ;6576
                                    ;6577   PA.88.A:
                                    ;6578           ;11011-------------------------;CHANGE #2:SAVE DEST CONTENTS IN [TO]
                                    ;6579           RC[TO]_LA,ALU?,J/ADDFDX,        ;CONVERT SC FROM EX256 TO EX128
U 12B6, 0000,1B3C,9D80,F980,0084,A13E ;6580         SC_SC-K[.7C]                    ;PLUS UNKNOWN CORRECTION
                                    ;6581
                                    ;6582   PA.MULFX:
                                    ;6583           ;-----------------------------;
                                    ;6584           D_DAL.SC,                       ;SHIFT M'IER READY, R[R15]_M'CAND
                                    ;6585           R[R15]_LC.RIGHT,SI/ZERO,        ;
U 12BA, 0D50,0038,8584,FAF8,0084,70F0 ;6586         SC_K[.C],SD_SS                  ;SET LOOP CT FOR 26. BITS
                                    ;6587
                                    ;6588   =0*     ;0*----------------------------;
                                    ;6589           ALU_0(A),                       ;L3_M'CAND
                                    ;6590           D_D.RIGHT2,SI/ZERO,LAB_R[R15],  ;
U 10F0, 0203,0C3D,0180,FA78,0000,0350 ;6591         CALL,BEN/MUL,J/MULPP            ;CALL MULTIPLICATION ROUTINE
                                    ;6592
                                    ;6593           ;1*----------------------------;
U 10F2, 0F00,003C,01E0,F800,0081,12BE ;6594         SC_FE,D_Q,Q_D                   ;ALWAYS POS PROD
                                    ;6595
                                    ;6596           ;-----------------------------;
                                    ;6597           SC_SC-SHF.VAL,D_DAL.NORM,       ;SHIFT LEFT JUSTIFIED
U 12BE, 0E00,003C,4180,F800,008C,B2D0 ;6598         K[.80]                          ;
                                    ;6599
                                    ;6600           ;-----------------------------;
                                    ;6601           D_D+K[.80],CLK.UBCC,            ;ROUNDING
U 12D0, 0819,0014,4180,F858,0010,12B6 ;6602         LAB_R(PRN),J/PA.88.A            ;CHANGE#1:PREFETCH CONTENTS OF DEST
```

```
                                    ;6603   ;Date    : 24-Jul-80(18-Dec-80)                                  Patch no. 089
                                    ;6604   ;Author : Robert J. Avarbock
                                    ;6605   ;
                                    ;6606   ;Source code file  : DECMAL
                                    ;6607   ;First included in : WCS123
                                    ;6608   ;
                                    ;6609   ;PROM address to be intercepted : 07FA
                                    ;6610   ;
                                    ;6611   ;Description of problem :
                                    ;6612   ;       ADDP4/6 SUBP4/6 will not set the V bit properly when it
                                    ;6613   ;       has dst len of <n> and answer of -10^<n>
                                    ;6614   ;       I.E. dst len =3, result =-1000
                                    ;6615   ;
                                    ;6616   ;Instead of that microinstruction, insert here the code which works :
                                    ;6617
                                    ;6618   1199:   ;-------------------------------;
U 1199, 0000,1A3C,0180,F800,0000,123B ;6619         PSL.Z?                          ;  If Z set, then not all 9's...
                                    ;6620
                                    ;6621   =1011   ;1011---------------------------;  Z not set.
U 123B, 0000,123C,0180,F800,0000,02AA ;6622         EALU?,J/ADDSUB                  ;  Back to PCS.
                                    ;6623
                                    ;6624           ;1111---------------------------;  Z set - potential no OV.
                                    ;6625           STATE_STATE.OR.K[.20],          ;  Set not all 9's bit.
U 123F, 0000,123C,7580,F800,1404,22AA ;6626         EALU?,J/ADDSUB                  ;
```

```
                                ;6627   ;Date   : 2-Aug-80                                                  Patch no. 090
                                ;6628   ;Author : Robert J. Avarbock
                                ;6629   ;
                                ;6630   ;Source code file  : DECMAL
                                ;6631   ;First included in : WCS123
                                ;6632   ;
                                ;6633   ;PROM address to be intercepted : 0BB2
                                ;6634   ;
                                ;6635   ;Description of problem :
                                ;6636   ;       On start-up of CVTTP, the STATE register is not
                                ;6637   ;       cleared or initialized before the first potential
                                ;6638   ;       memory access.  If a memory management trap occurs on this
                                ;6639   ;       first reference, if the interrupt bit of STATE was left from
                                ;6640   ;       the previous use of STATE, the Ucode will go try to
                                ;6641   ;       service a non-existant interrupt instead of memory management.
                                ;6642   ;       That first reference was reading the trailing byte from
                                ;6643   ;       the source string.
                                ;6644   ;
                                ;6645   ;Instead of that microinstruction, insert here the code which works :
                                ;6646
                                ;6647   119A:   ;--------------------------------;
                                ;6648           FE_K[.14],                      ;  SC GETS SRC-LENGTH-1, FE GETS 20.
                                ;6649           ALU_0-Q-1,R[R2]_ALU,            ;  INIT DEST-LENGTH.
                                ;6650           STATE_K[.14],                   ;  *** PATCH ***  (BIT .80 IS CLEAR)
U 119A, 001F,0008,21B0,FA90,1504,6631   ;6651           QK/RIGHT,J/T2P.X1               ;  Back to PCS.
```

```
                                ;6652   ;Date    : 14-Aug-80                                                Patch no. 091
                                ;6653   ;Author : Robert J. Avarbock
                                ;6654   ;
                                ;6655   ;Source code file  : PROBE
                                ;6656   ;First included in : WCS123
                                ;6657   ;
                                ;6658   ;PROM address to be intercepted : 0D06
                                ;6659   ;
                                ;6660   ;Description of problem :
                                ;6661   ;       The PROBE instructions clear the 'C' bit when it should
                                ;6662   ;       Keep it's previous value.
                                ;6663   ;
                                ;6664   ;Instead of that microinstruction, insert here the code which works :
                                ;6665
                                ;6666   119B:   ;-------------------------------;
                                ;6667           D_DAL.SC,                       ;PREVIOUS MODE TO D1-0
U 119B, 0D19,2024,85C0,F800,0000,12D2   ;6668   Q_Q.ANDNOT.K[.C]                ;Save PSL except for N & Z here.
                                ;6669
                                ;6670           ;-------------------------------;
                                ;6671           RC[T3]_Q.ANDNOT.K[.2],          ;  Also clear V.
U 12D2, 0019,2024,0980,F998,0000,0D08   ;6672   J/PROB.X                        ;  Return to PCS.
```

```
                                ;6673   ;Date   : 15-Aug-80                                                  Patch no. 092
                                ;6674   ;Author : Robert J. Avarbock
                                ;6675   ;
                                ;6676   ;Source code file  : DECMAL
                                ;6677   ;First included in : WCS123
                                ;6678   ;
                                ;6679   ;PROM address to be intercepted : 074E
                                ;6680   ;
                                ;6681   ;Description of problem :
                                ;6682   ;       CVTPT Does not set the V bit when the only non-zero data
                                ;6683   ;       is in the sign byte.
                                ;6684   ;
                                ;6685   ;Instead of that microinstruction, insert here the code which works :
                                ;6686
                                ;6687   119C:   ;1*-------------------------------;
                                ;6688           SC_K[.8],                       ;  FOR LATER SHIFTING
U 119C, 0013,8014,0180,F800,0284,72D4   ;6689           ALU_D.0XT[BYTE]+LC,VAK/LOAD     ;  LOAD INDEXED TABLE-ADDRESS
                                ;6690   =;END
                                ;6691           ;-------------------------------;  BR ON SIGN NIB.
                                ;6692           ALU_D.0XT[BYTE].AND.K[.FFF0],   ;  Punt sign nibble.
U 12D4, 001B,8F34,6D80,F998,0000,122A   ;6693           RC[T3]_ALU,BCDSGN?              ;  Save possible OV.  BR on sign.
                                ;6694
                                ;6695   =10     ;10-------------------------------;  BR ON SIGN NIB.
                                ;6696           ALU_D.0XT[BYTE].AND.K[.FFF0],   ;  STRIP OFF SIGN-NIBBLE
                                ;6697           LC_RC[T1],N_AMX.Z_TST,          ;  GET DST-LENGTH, CLK PSL<Z>
                                ;6698           D[BYTE]_CACHE,Q_0,              ;  READ TABLE-ENTRY
U 122A, 001B,8034,6DF8,4108,0030,12D6   ;6699           J/PA92.0                        ;
                                ;6700
                                ;6701           ;11-------------------------------;
                                ;6702           STATE_STATE+1,                  ;  SET MINUS-BIT OF STATE
                                ;6703           ALU_D.0XT[BYTE].AND.K[.FFF0],   ;  STRIP OFF SIGN-NIBBLE
                                ;6704           LC_RC[T1],N_AMX.Z_TST,          ;  GET DST-LENGTH, CLK PSL<Z>
U 122B, 001B,8034,6DF8,4108,1430,D2D6   ;6705           D[BYTE]_CACHE,Q_0               ;  READ TABLE-ENTRY
                                ;6706   =;END
                                ;6707
                                ;6708           ;-------------------------------;
                                ;6709   PA92.0: D_DAL.SC,Q_D,                    ;  STORE RESULT IN D AND Q
                                ;6710           ALU_0+LC+1,SC_ALU,              ;  STORE DST-LENGTH IN SC
U 12D6, 0D13,0010,01E0,F800,0092,12D8   ;6711           CLK.UBCC                        ;
                                ;6712
                                ;6713           ;-------------------------------;
                                ;6714           D_Q,Q_D,                        ;  D GETS DATA IN BYTE 0
                                ;6715           SC_SC+1,                        ;  INCREMENT DST-COUNT
                                ;6716           ALU_R[R3],VAK/LOAD,             ;  LOAD DST-ADDRESS
U 12D8, 0C00,013C,01E0,FA18,0280,D0CC   ;6717           Z?,J/PA92.1                     ;  TEST DST-LENGTH
                                ;6718
                                ;6719   =0     ;0-------------------------------;  BRANCH ON ALU Z-BIT
                                ;6720   PA92.1: EALU_FE,CLK.UBCC,                ;  CLOCK SRC-COUNT
                                ;6721           ALU_0(A),RC[T3]_ALU,            ;  No ov data yet.
U 10CC, 00C3,023C,0180,F998,0010,6BE6   ;6722           ROR?,J/P2NI1                    ;  TEST DST-ADDRESS FOR WORD ALIGNMENT
                                ;6723
                                ;6724           ;1-------------------------------;
                                ;6725           ALU_0(A),LAB_R1&RC[T2]_ALU,     ;  NO LEFT-OVER DIGIT
                                ;6726           EALU_FE,CLK.UBCC,               ;  CLOCK SRC-LENGTH
U 10CD, 0003,003C,0180,FB10,0010,6BF6   ;6727           J/P2NL                          ;  JUMP TO MAIN-LOOP.
```

```
                                        ;6728   ;Date    : 22-Aug-80                                                    Patch no. 093
                                        ;6729   ;Author : Robert J. Avarbock
                                        ;6730   ;
                                        ;6731   ;Source code file  : DECMAL
                                        ;6732   ;First included in : WCS123
                                        ;6733   ;
                                        ;6734   ;PROM address to be intercepted : 0C47
                                        ;6735   ;
                                        ;6736   ;Description of problem :
                                        ;6737   ;       (ADD,SUB)P(4,6) won't set PSL<v> if overflow data
                                        ;6738   ;       is at least a longword of all 9's.
                                        ;6739   ;
                                        ;6740   ;Instead of that microinstruction, insert here the code which works :
                                        ;6741
                                        ;6742   119D:   ;--------------------------------;
                                        ;6743           STATE_STATE.OR.K[.50],          ;  SET NEGATE AND OV BITS.
                                        ;6744           ALU_R[R15],D_ALU.RIGHT,SI/ASHR, ;  DIVIDE LEN BY 2.
                                        ;6745           CLK_UBCC,LONG,IR0?,             ;  TEST FOR 4 OR 6 OPRS.
U 119D, 0840,1B3C,3480,FA78,1414,299D   ;6746           J/NEGA0                         ;
```

```
                                        ;6747   ;Date    : 09-Jan-81,(05-Jun-81)                                  Patch no. 094
                                        ;6748   ;Author : Robert J. Avarbock
                                        ;6749   ;
                                        ;6750   ;Source code file  : REI
                                        ;6751   ;First included in : WCS123
                                        ;6752   ;
                                        ;6753   ;PROM address to be intercepted : 0EDB
                                        ;6754   ;
                                        ;6755   ;Description of problem :
                                        ;6756   ;       When an REI is used to enter compatibility mode, if we
                                        ;6757   ;       are going to an odd address, the hardware does things so
                                        ;6758   ;       that the PC that gets reported back via the exception, is
                                        ;6759   ;       trash.  This patch will have the exception handler return
                                        ;6760   ;       the address of the offending odd address.
                                        ;6761   ;
                                        ;6762   ;Instead of that microinstruction, insert here the code which works :
                                        ;6763
                                        ;6764   11A4:    ;-------------------------------;
U 11A4, 0010,0038,0180,F910,0010,12DA   ;6765           ALU_RC[T2],CLK.UBCC             ;  Get PSL and set ALU<N> on CM.
                                        ;6766
                                        ;6767           ;-------------------------------;
U 12DA, 0010,1B38,0180,F909,0200,1077   ;6768           PC&VA_RC[T1],ALU.N?             ;  Get rude odd addr., br on CM.
                                        ;6769
                                        ;6770   =0111   ;0111-------------------------;  *** Not compatibility mode ***
                                        ;6771           PC&VA_RC[T1],FLUSH.IB,          ;  Set new PC, and start IB.
U 1077, 2010,1938,0180,F909,4200,0BD8   ;6772           D2-0?,J/REI.12                  ;  Check new IS and mode in PCS.
                                        ;6773
                                        ;6774           ;1111-------------------------;  *** Entering compatibility mode ***
U 107F, 0000,153C,0180,F800,0000,124E   ;6775           ALU1-0?                         ;  See if odd address.
                                        ;6776
                                        ;6777   =1110   ;1110-------------------------;  *** Even address - all is well. ***
                                        ;6778           PC&VA_RC[T1],FLUSH.IB,          ;  Set new PC, and start IB.
U 124E, 2010,1938,0180,F909,4200,0BD8   ;6779           D2-0?,J/REI.12                  ;  Check new IS and mode in PCS.
                                        ;6780
                                        ;6781           ;1111-------------------------;  *** Odd address ***
                                        ;6782           STATE_0(A),                     ;  Clear state for later.
U 124F, 001B,0038,D580,F988,1408,72DC   ;6783           RC[T1]_K[.6]                    ;  Comp mode odd address trap.
                                        ;6784
                                        ;6785           ;-------------------------------;
                                        ;6786           RC[T0]_K[.30],                  ;  Comp mode vector.
U 12DC, 0018,0038,7980,F980,0000,08E0   ;6787           J/IE.PA.94                      ;  Back into the exception.
```

```
                                ;6788   ;Date   : 28-Jan-80                                                  Patch no. 095
                                ;6789   ;Author : Robert J. Avarbock
                                ;6790   ;
                                ;6791   ;Source code file  : CHAR
                                ;6792   ;First included in : WCS123
                                ;6793   ;
                                ;6794   ;PROM address to be intercepted : 06E4
                                ;6795   ;
                                ;6796   ;Description of ECO :
                                ;6797   ;       MOVC(3,5) should make use of the move quad extended write
                                ;6798   ;       function of the 11/780.  When doing fill, at best the Ucode
                                ;6799   ;       does longword writes.  Add code to do quadword writes when
                                ;6800   ;       destination is quad-aligned.  Extended write function writes
                                ;6801   ;       the contents of the D-register twice.
                                ;6802
                                ;6803   119E:   ;-------------------------------;
U 119E, 0019,2034,A180,F800,0010,12DE   ;6804           ALU_Q.AND.K[.FFE0],CLK.UBCC     ;  Punt if less than 32 bytes.
                                ;6805
                                ;6806           ;-------------------------------;
                                ;6807           D_R[R3],K[.7],                  ;  Dst. addr.
U 12DE, 0800,013C,5D80,FA18,0000,10D4   ;6808           Z?,J/PA95.0                     ;  If Z set, then < 32 bytes of fill.
                                ;6809   =0
                                ;6810   PA95.0: ;0-------------------------------;  Z not set - Continue this code.
                                ;6811           D_Q-K[.7]-1, RC[T2]_ALU,        ;  Assume 8 bytes left, save new count.
                                ;6812           C[K.UBCC,                       ;  Set C31 if 8 or more bytes left.
                                ;6813           STATE_STATE.ANDNOT.K[.7],       ;  Assume quad or long Xfer.
U 10D4, 0819,3908,5D80,F990,1414,525B   ;6814           D3?,J/PA95.1                    ;  Br on Dst addr low bits, for type.
                                ;6815
                                ;6816           ;1-------------------------------;  Z set - code not worth doing.
                                ;6817           D_Q-K[.3]-1,RC[T2]_ALU,         ;  Assume 4 bytes left, save new cnt.
                                ;6818           STATE_STATE.ANDNOT.K[.3],       ;  Assume longword Xfer,
U 10D5, 0819,2008,0D80,F990,1414,50DC   ;6819           CLK.UBCC,J/PA95.2               ;
                                ;6820
                                ;6821   =1011
                                ;6822   PA95.1: ;1011-----------------------------;  *** Quad fill. ***
                                ;6823           LAB_R[R1],                      ;
U 125B, 0000,033C,0180,FA08,0000,10F4   ;6824           C31?,J/PA95.Q                   ;  Go see if count big enough.
                                ;6825
                                ;6826           ;1111-----------------------------;  *** Longword fill. ***
                                ;6827           D_Q-K[.3]-1,RC[T2]_ALU,         ;  Assume 4 bytes left, save new cnt.
                                ;6828           STATE_STATE.ANDNOT.K[.3],       ;  Assume longword Xfer,
U 125F, 0819,2008,0D80,F990,1414,50DC   ;6829           CLK.UBCC,J/PA95.2               ;
                                ;6830
                                ;6831   PA95.8: ;-------------------------------;
                                ;6832           D_Q-K[.3]-1,RC[T2]_ALU,         ;  Assume 4 bytes left, save new cnt.
                                ;6833           STATE_STATE.ANDNOT.K[.3],       ;  Assume longword Xfer,
U 12E2, 0819,2108,0D80,F990,1414,50DC   ;6834           Z?,CLK.UBCC,J/PA95.2            ;  Br on zero count.
                                ;6835
                                ;6836   =0
                                ;6837   PA95.2: ;0-------------------------------;  When branching, here if count<>0.
                                ;6838           LAB_R[R1],Q_D,                  ;  Definitely a longword.
U 10DC, 0000,033C,01E0,FA08,0000,05EC   ;6839           C31?,J/MOVCFILLMORE             ;
                                ;6840
                                ;6841           ;1-------------------------------;  Zero count left - all done.
U 10DD, 0003,003C,0180,FA80,0000,09EE   ;6842           R[R0]_0,J/R245ZERO              ;  See y'all!
```

```
                                        ;6843   =0*
                                        ;6844   PA95.Q: ;0*------------------------------;  Quad doesn't fit, back and try long.
                                        ;6845           D_Q-K[.3]-1,RC[T2]_ALU,         ;  Assume 4 bytes left, save new cnt.
                                        ;6846           STATE_STATE.ANDNOT.K[.3],       ;  Assume longword Xfer,
                                        ;6847           CLK.UBCC,                       ;  Set C31 on this.
U 10F4, 0819,2008,0D80,F990,1414,50DC   ;6848           J/PA95.2                        ;
                                        ;6849
                                        ;6850           ;1*------------------------------;  Quad fits! See if MM on.
U 10F6, 0C00,003C,49F0,2C00,0000,12E4   ;6851           Q_ID[TBERO],D_Q                 ;  Bit <0> is MME bit.  Hold cnt in D.
                                        ;6852
                                        ;6853   PA95.9: ;------------------------------;
U 12E4, 0001,203C,0180,F800,0000,12E6   ;6854           ALU_Q                           ;  Set ALU<0> for me.
                                        ;6855
                                        ;6856           ;------------------------------;  BR on ALU<1:0>
U 12E6, 0810,1538,01E0,F910,0000,126E   ;6857           Q_D,D_RC[T2],ALU1-0?            ;  Restore things, see if MM on.
                                        ;6858
                                        ;6859   =1110   ;1110------------------------------;  MM off - punt this stuff.
                                        ;6860           D_Q-K[.3]-1,RC[T2]_ALU,         ;  Assume 4 bytes left, save new cnt.
                                        ;6861           STATE_STATE.ANDNOT.K[.3],       ;  Assume longword Xfer,
                                        ;6862           CLK.UBCC,                       ;  Set C31 on this.
U 126E, 0819,2008,0D80,F990,1414,50DC   ;6863           J/PA95.2                        ;
                                        ;6864
                                        ;6865           ;1111------------------------------;  MM on - proceed.
                                        ;6866           LA_RA[3],VA_LA,FE_SC,           ;  Do a dummy LW-write to set PTE<M>
U 126F, 0000,003C,01E0,F898,0300,12F2   ;6867           Q_D                             ;  Update count.  (CNT-8)
                                        ;6868
                                        ;6869           ;------------------------------;
                                        ;6870           RC[T2]_Q+K[.4],                 ;  Fix size in case of a fault.
U 12F2, 0019,2014,1180,3190,0000,12FD   ;6871           CACHE_D[LONG]                   ;  Do the write.
                                        ;6872
                                        ;6873           ;------------------------------;  *** Here to go off and get PTE.
U 12FD, 0C14,0038,B980,3DE0,0000,130E   ;6874           ID[D.SV]_D,RC[PC.SV]_PC,D_Q     ;  We must save D,PC
                                        ;6875
                                        ;6876           ;------------------------------;
U 130E, 0010,0038,BD80,3DC0,0000,12B0   ;6877           ID[Q.SV]_D,RC[LC.SV]_LC         ;  And Q and LC and SC next Uorder.
                                        ;6878
                                        ;6879   =00     ;00------------------------------;  Guarantee alignment for RETURN.
                                        ;6880           MEMORY.NOP,SET.NEST.ERR,        ;  Freeze mem, and set PC saved flag.
                                        ;6881           PC_VA,RC[SC.SV]_K[SC],          ;  Routine wants VA in PC, save SC.
U 12B0, 0018,1C39,1D80,09E9,2C00,023C   ;6882           CALL,PSL.MODE?,J/GET.PTE        ;  Go fetch the PTE.
                                        ;6883
                                        ;6884           ;01------------------------------;  *** Good return ***  PTE in D.
                                        ;6885           Q_K[.3FF].RIGHT,FE_SC,          ;  D,PC,Q,LC Must be restored...
U 12B1, 0058,0038,81C0,F800,0100,1311   ;6886           J/PA95.3                        ;  Go calc PHYS addr.
                                        ;6887
                                        ;6888           ;10------------------------------;  Oops - loser.
                                        ;6889           MEMORY.NOP,RC[VA.SV]_PC,        ;  Go away unhappy.
U 12B2, 0014,1138,0180,09C8,0000,0D??   ;6890           LAST.REF?,J/M.FLT               ;
```

```
                                    ;6891   =;      Here to remove PFN from PTE...
                                    ;6892   PA95.3: ;-------------------------------;  *** D,PC,Q,PC,SC Still saved.
U 1311, 000D,2034,D9C0,FA18,0084,7319 ;6893         Q_Q.AND.R[R3],SC_K[.9]          ;  Isolate LSB's.  Bits to shift PFN
                                    ;6894
                                    ;6895           ;-------------------------------;
U 1319, 0D10,0038,0180,F961,0200,131E ;6896         PC&VA_RC[PC.SV],D_DAL.SC        ;  D now has PFN shifted to proper place.
                                    ;6897
                                    ;6898           ;-------------------------------;
U 131E, 0100,003C,0180,FA08,0100,1322 ;6899         D_D.LEFT2,FE_SC,LAB_R[R1]       ;  Restore LB with R1.
                                    ;6900
                                    ;6901           ;-------------------------------;
U 1322, 0201,203C,0180,F800,0082,1329 ;6902         D_D.RIGHT2,SC_Q                 ;  Clr out upper two bits. Keep Low 9.
                                    ;6903
                                    ;6904           ;-------------------------------;
U 1329, 081D,0014,0180,F800,0200,1331 ;6905         D_D+Q,VAK/LOAD                  ;  D & VA now have correct phys addr.
                                    ;6906
                                    ;6907           ;-------------------------------;
U 1331, 0800,003C,0180,F8A8,0000,1339 ;6908         LA_RA[5],D_LA                   ;  Data to write.
                                    ;6909
                                    ;6910           ;-------------------------------;  Restore things...
                                    ;6911           Q_ID[Q.SV],                     ;
                                    ;6912           LA_RA[3],INTRPT.STROBE,         ;  Strobe for int's
U 1339, 0000,003C,BDF0,2C98,4000,1106 ;6913         J/PA95.W                        ;  Do the write.
                                    ;6914
                                    ;6915
                                    ;6916   =0*
                                    ;6917   PA95.7: ;0*-----------------------------;  Less than a quad left.
U 1104, 0019,2014,01C0,F800,0010,12E2 ;6918         Q_Q+K[.8],CLK.UBCC,J/PA95.8     ;  Restore Q, go see if zero.
                                    ;6919
                                    ;6920   PA95.W: ;1*-----------------------------;  Quad fits..
                                    ;6921           CACHE_D(QUAD),                  ;  *** The big write ***
                                    ;6922           R[R3]_LA+K[.8],BEN/INTERRUPT,   ;  Increment address.
U 1106, 0018,0E14,0180,A29B,0084,9296 ;6923         SC_SC+K[.8],VA_VA+4,J/PA95.W1   ;
                                    ;6924
                                    ;6925   =110
                                    ;6926   PA95.W1:;110----------------------------;  No int.
                                    ;6927           D_Q-K[.8],CLK.UBCC,RC[T2]_ALU,  ;  Update count.
U 1296, 0819,3400,018C,F993,0010,12B5 ;6928         SC?,VA_VA+4,J/PA95.5            ;  VA now has been inc'd by 8.
                                    ;6929
                                    ;6930           ;111----------------------------;  Int pending.
U 1297, 0C00,003C,0180,F800,0081,0733 ;6931         SC_FE,D_Q,J/MOVCPACKST          ;  Put count in D, go service.
                                    ;6932
                                    ;6933   =101
                                    ;6934   PA95.5: ;101----------------------------;  SC <=0 - page boundary hit.
U 12B5, 0CC0,003C,01E0,F800,0000,1342 ;6935         D_Q,Q_D,J/PA95.P                ;  Go see if done before regetting PTE.
                                    ;6936
                                    ;6937           ;111----------------------------;  Still on same page.
U 12B7, 0800,003C,01E0,F8A8,0000,133D ;6938         Q_D,LA_RA[5],D_LA               ;  Update count, get data.
                                    ;6939
                                    ;6940           ;-------------------------------;  Same page.
                                    ;6941           LA_RA[3],                       ;  Replace count, get addr, get data.
U 133D, 0000,033C,0180,F898,4000,1104 ;6942         C3T?,INTRPT.STROBE,J/PA95.7     ;  Go do write branching on count.
                                    ;6943
                                    ;6944   PA95.P: ;1----------------------------;  Page boundary being hit.
U 1342, 0C00,0D3C,01E0,F800,0000,12E1 ;6945         D_Q,Q_D,SIGNS?,J/PA95.B         ;  Set up to check count.
```

```
                                        ;6946   ;       *** Here with D=Q-8.  we swapped them to use the BEN/SIGNS
                                        ;6947   ;       *** and then swapped them back to keep things from crashing...
                                        ;6948
                                        ;6949   =001
                                        ;6950   PA95.B:
                                        ;6951   =011    ;011--------------------------;  D was .GE. 0, Q was <> 0.
U 12E3, 0C00,003C,49F0,2C00,0000,12E4   ;6952           Q_ID[TBERO],D_Q,J/PA95.9       ;  Go get PTE again.
                                        ;6953
                                        ;6954           ;101--------------------------;  D was neg, Q was =0
U 12E5, 0003,003C,0180,FA80,0000,09EE   ;6955           R[RO]_0,J/R245ZERO             ;  All done.
                                        ;6956
                                        ;6957           ;111--------------------------;  D was neg, Q was <> 0
                                        ;6958           D_Q-K[.3]-1,RC[T2]_ALU,        ;  Go back and do 4.
                                        ;6959           STATE_STATE.ANDNOT.K[.3],      ;
U 12E7, 0819,2008,0D80,F990,1414,50DC   ;6960           CLK.UBCC,J/PA95.2              ;
```

```
                                ;6961   ;Date   : 28-Jan-81                                                  Patch no. 096
                                ;6962   ;Author : Robert J. Avarbock
                                ;6963   ;
                                ;6964   ;Source code file  : MOVPR
                                ;6965   ;First included in : WCS123
                                ;6966   ;
                                ;6967   ;PROM address to be intercepted : 0564
                                ;6968   ;
                                ;6969   ;Description of ECO :
                                ;6970   ;       Add processor register 3F (hex).
                                ;6971   ;       Instruction to test whether an address produces a TB MISS or not.
                                ;6972   ;       We are now at the point of knowing that we have an illegal
                                ;6973   ;       processor register.  Before doing anything, we must check to see
                                ;6974   ;       if we really are using ^X3F.  I decided to do it this way so that
                                ;6975   ;       future uses of illegal processor registers will not require additional
                                ;6976   ;       traps.  Once we decide that it is ^X3F, the addr will be in D.
                                ;6977   ;       If there is a TB HIT, ALU<V> will be set, otherwise cleared.
                                ;6978   ;
                                ;6979
                                ;6980
                                ;6981   1159:   ;-------------------------------;
U 1159, 0001,003C,5580,F800,0200,1344   ;6982           VA_D,K[.3F]                     ;  Get his addr into VA, Get 3F.
                                ;6983
                                ;6984           ;-------------------------------;
U 1344, 0C00,003C,5580,F800,0014,B345   ;6985           EALU_SC-K[.3F],CLK.UBCC         ;  If zero left, then it was PRN ^X3F.
                                ;6986
                                ;6987           ;-------------------------------;
                                ;6988           ALU_D,N&Z_ALU.V&C_0,            ;  Set ALU<N,Z> on src, Clr PSL<V>
                                ;6989           TEST.TB.RCHK,                   ;  Clock potential TB miss..
U 1345, 0001,123C,0180,0000,0050,127B   ;6990           EALU.Z?                         ;  If EALU<Z> set, then it was PRN ^X3F.
                                ;6991
                                ;6992   =1011   ;1011-------------------------;  Not PRN ^X3F - Give bad PRN return.
U 127B, 0000,003E,0180,F800,0000,0010   ;6993           RETURN10                        ;  Too bad...
                                ;6994
                                ;6995           ;1111-------------------------;  It was PRN ^X3F - continue on.
U 127F, 0000,1D3C,0180,0600,0000,109C   ;6996           INHIBIT.IB,TB.TEST?             ;  Branch on TB-MISS.
                                ;6997
                                ;6998   =11100  ;11100------------------------;  *** TB HIT ***
U 109C, 0000,003E,0180,F800,0020,0018   ;6999           SET.V,RETURN18                  ;  Set PSL<V> and return.
                                ;7000
                                ;7001           ;11101------------------------;  *** 1st Modify to page ***
U 109D, 0000,003E,0180,F800,0020,0018   ;7002           SET.V,RETURN18                  ;  Set PSL<V> and return.
                                ;7003
                                ;7004           ;11110------------------------;  *** Protection Violation ***
U 109E, 0000,003E,0180,F800,0020,0018   ;7005           SET.V,RETURN18                  ;  Set PSL<V> and return.
                                ;7006
                                ;7007           ;11111------------------------;  *** TB MISS ***
U 109F, 0000,003E,0180,F800,0000,0018   ;7008           RETURN18                        ;  PSL<V> still clear.  Return.
```

```
                                        ;7009   ;Date   : 10-Mar-81                                          Patch no. 097
                                        ;7010   ;Author : Robert J. Avarbock
                                        ;7011   ;
                                        ;7012   ;Source code file  : EDIT
                                        ;7013   ;First included in : WCS123
                                        ;7014   ;
                                        ;7015   ;PROM address to be intercepted : 0B0D
                                        ;7016   ;
                                        ;7017   ;Description of problem :
                                        ;7018   ;       On a reserved operand fault because of an unimplemented
                                        ;7019   ;       pattern operator, EDITPC does not pack itself up to be
                                        ;7020   ;       restarted if the programmer requests a restart.  The instruction
                                        ;7021   ;       unpacks junk and does who knows what...
                                        ;7022   ;
                                        ;7023   ;Instead of that microinstruction, insert here the code which works :
                                        ;7024
                                        ;7025   11A5:   ;--------------------------------;
                                        ;7026           D_Q,Q_D,                        ;  For left shift Q'D concat.
U 11A5, 0C18,0038,7DE0,F980,0000,1346   ;7027           R[[T0]_K[.18]                   ;  Vector ID for RESOPR.
                                        ;7028
                                        ;7029           ;--------------------------------;
                                        ;7030           D_DAL.SC,                       ;  D<31:24>=SXT, <23:16>=ADJ INP CNTR,
                                        ;7031           LAB_R[R1],                      ;  D<15:7>=0, D<7:0>=SC LEN.
U 1346, 0D00,003C,5980,FA08,1404,5349   ;7032           STATE_STATE.AN.6T04             ;  Clear out all but <7>.
                                        ;7033
                                        ;7034           ;--------------------------------;
U 1349, 0001,003C,0180,FA80,0000,134C   ;7035           R[R0]_D                         ;  Save current SRC count stuff.
                                        ;7036
                                        ;7037           ;--------------------------------;  Flag restart code we came from here.
U 134C, 0818,0030,4180,FA20,0000,134D   ;7038           D_R[R4].OR.K[.80]               ;  Save original count. (80 for 0 len)
                                        ;7039
                                        ;7040           ;--------------------------------;
U 134D, 0B00,003C,7180,F800,0084,7350   ;7041           SC_K[.FFF8],D_D.SWAP            ;  Simulate EDITFPD. D has bytes 1 0 0 0
                                        ;7042
                                        ;7043           ;--------------------------------;
U 1350, 0D18,0034,C1C0,FA10,0000,1352   ;7044           D_DAL.SC,Q_R[R2].AND.K[.FFFF]   ;  D has bytes Q<7:0> 1 0 0
                                        ;7045
                                        ;7046           ;--------------------------------;
U 1352, 001D,2030,0580,FA90,1484,3281   ;7047           R[R2]_Q.OR.D,SC_STATE.OR.K[.1]  ;  R2 has Q<7:0> 1 Q<15:8> Q<7:0>
```

```
                                    ;7048           ;Patch no. 097 continued...
                                    ;7049
                                    ;7050   =01     ;01-----------------------------;  Simulate FPDPACK, only use R4.
U 1281, 0014,0039,01C0,F800,0000,0EB8 ;7051         Q_PC,CALL,J/BAKUP.PC            ;  Get PC delta.
                                    ;7052
                                    ;7053           ;11-----------------------------;  Return2 from BAKUP.PC
U 1283, 0B01,203C,0180,F9E0,0000,1354 ;7054         RC[PC.SV]_Q,D_D.SWAP            ;  PC.SV may be needed in MM.
                                    ;7055
                                    ;7056           ;-------------------------------;
U 1354, 0018,0038,1DC0,F800,0000,1356 ;7057         Q_SC                            ;  Q gets STATE.
                                    ;7058
                                    ;7059           ;-------------------------------;
U 1356, 0000,003C,7180,F800,0084,735B ;7060         SC_K[.FFF8]                     ;  Magic shift count.
                                    ;7061
                                    ;7062           ;-------------------------------;
U 135B, 0D18,0034,C1C0,FA00,0000,135D ;7063         D_DAL.SC,Q_R[R0].AND.K[.FFFF]   ;  Keep low word.  D has STATE & PC del.
                                    ;7064
                                    ;7065           ;-------------------------------;
U 135D, 001D,0030,0180,FAA0,0000,1360 ;7066         R[R4]_D.OR.Q                    ;  R4 now has packed stuff.
                                    ;7067
                                    ;7068           ;-------------------------------;
U 1360, 0000,163C,0180,FA08,0000,1031 ;7069         LAB_R[R1],STATE7-4?             ;  Set up LA for R1 alignment.
                                    ;7070
                                    ;7071   =0***   ;1***-----------------------------;  R1 is ok.
U 1031, 0000,003C,0180,F800,2400,08FE ;7072         SET.FPD,J/EXCPTO                ;  Go take exception.
                                    ;7073
                                    ;7074           ;1***-----------------------------;  R1 needs increment.
U 1039, 0018,0014,0580,FA88,2400,08FE ;7075         R[R1]_LA+K[.1],SET.FPD,J/EXCPTO ;  Go take exception.
```

```
                                ;7076   ;Date    : 10-Mar-81                                                    Patch no. 098
                                ;7077   ;Author  : Robert J. Avarbock
                                ;7078   ;
                                ;7079   ;Source code file  : EDIT
                                ;7080   ;First included in : WCS123
                                ;7081   ;
                                ;7082   ;PROM address to be intercepted : 0B1D
                                ;7083   ;
                                ;7084   ;Description of problem :
                                ;7085   ;       Patch no. 097 packes up EDITPC for RESOPR restarts.  This
                                ;7086   ;       patch unpacks from R4 as packed from patch 097 and restarts
                                ;7087   ;       the instruction PROPERLY.
                                ;7088   ;
                                ;7089   ;Instead of that microinstruction, insert here the code which works :
                                ;7090
                                ;7091   11A6:   ;-------------------------------;
                                ;7092           R[T1]_Q.AND.K[.FF],             ;  Save fill char.
U 11A6, 0B19,3834,4980,F988,0081,128B   ;7093           SC_FE,SWAPD,D.BYTES?            ;  Unpack stuff, check exp flag (pat 98)
                                ;7094
                                ;7095   =1011   ;1011---------------------------;  Flag not set - R0 unpack valid.
                                ;7096           D_DAL.SC,Q_D.AND.K[.FF],        ;  Rejoin PCS unharmed.
U 128B, 0D19,0034,49C0,FA10,0000,0B22   ;7097           LAB_R[R2],J/ED.PA.98            ;
                                ;7098
                                ;7099           ;1111---------------------------;  Flag set - unpack R4.
U 128F, 0800,003C,6DC0,FA20,0084,7362   ;7100           D_R[R4],Q_R[R4],SC_K[.FFF0]     ;
                                ;7101
                                ;7102           ;-------------------------------;  Simulate FPDUNPACK.
U 1362, 0D19,2034,C180,FA80,0000,1368   ;7103           D_DAL.SC,R[R0]_Q.AND.K[.FFFF]   ;  Restore R0.
                                ;7104
                                ;7105           ;-------------------------------;
                                ;7106           Q_D.AND.K[.FF],                 ;  Get out PC delta.
U 1368, 0C19,0034,49C0,F800,0000,136A   ;7107           D_Q                             ;  D has what was saved at FPD time.
                                ;7108
                                ;7109           ;-------------------------------;
U 136A, 0B15,2014,6D80,F801,0304,736D   ;7110           C_D.SWAP,PC_Q+PC,FE_K[.FFF0]    ;  *** PC reset ***  -16 for shifting.
                                ;7111
                                ;7112           ;-------------------------------;
U 136D, 0001,003C,0180,FA28,0082,1370   ;7113           SC_D,LAB_R[R5]                  ;  STATE on its way home.
                                ;7114
                                ;7115           ;-------------------------------;
U 1370, 0018,0000,0580,FAA8,0000,1372   ;7116           R[R5]_LA-K[.1]                  ;  Move dst addr pointer back one.
                                ;7117
                                ;7118           ;-------------------------------;
                                ;7119           STATE_SC.VIA.KMX,               ;  State restored.
U 1372, 0800,003C,1DC0,FA10,1485,7374   ;7120           D_R[R2],Q_R[R2],SC_FE           ;  Set up to rejoin PCS.
                                ;7121
                                ;7122           ;-------------------------------;
U 1374, 0DC3,003C,0180,FAA0,0000,1376   ;7123           D_DAL.SC,R[R4]_0                ;  Extract 2nd MSByte for src len.
                                ;7124
                                ;7125           ;-------------------------------;
U 1376, 0C19,8034,5980,FAA0,0000,1378   ;7126           R[R4]_D.AND.K[.7F],BYTE,D_Q     ;  Get rid of MSB, restore D.
                                ;7127
                                ;7128           ;-------------------------------;
U 1378, 0B00,003C,0180,F800,0081,0B1E   ;7129           SC_FE,SWAPD,J/ED.PA.98.A        ;  Rejoin PCS to finish restoration.
```

```
                                        ;7130   .TOC    "FORKS.MIC"
                                        ;7131   .TOC    "Revision 1.3"
                                        ;7132   ;        P. R. Guilbault
                                        ;7133
;7134   .NOBIN
;7135   .REGION/0000,0FFF               ;Lower 4k for PCS
;7136
;7137   .TOC    "       Revision History"
;7138
;7139   ; 01    Remove absolute jumps.
;7140   ;       Change macro names that deal with condition codes.
;7141   ; 00    Create this file by merging AFORK.MIC, BFORK.MIC, and CFORK.MIC
;7142   ;       Start of history
;7143
                                        ;7144   .BIN
                                        ;7145   .NOLIST         ;Disable listing of PCS code for quickie assemblies
```

```
                                    ;7146   .TOC    ""      I-stream decode forks : A-FORK for VAX Instructions""
                                    ;7147
                                    ;7148   ;Control passes to this point from any "IRD" state.
                                    ;7149   ;The state of the data path is :
                                    ;7150   ;       LA          = Register selected by bits <3:0> of IB byte 2
                                    ;7151   ;       LB, VA, & D = Register selected by bits <3:0> of IB byte 1
                                    ;7152   ;       Q           = Instruction stream data, if any
                                    ;7153   ;       PC          = Address of next specifier
                                    ;7154
                                    ;7155   000:    ;-------------------------------;
                                    ;7156   A.FORK: D_Q,                           ;S^# SHORT LITERAL
U 0000, FC00,003F,01F0,F847,0000,0200 ;7157         B.FORK                          ;GO RIGHT TO NEXT EXECUTION POINT
                                    ;7158
                                    ;7159   001:    ;-------------------------------;RESERVED MODE
                                    ;7160   RSVMOD: RC[TO]_K[.7].LEFT2,            ;SETUP TRAP VECTOR ADDRESS
U 0001, 0078,0038,5D80,F980,0000,08FC ;7161         J/EXCPT                         ; AND TAKE A FAULT
                                    ;7162
                                    ;7163
                                    ;7164   002:    ;-------------------------------;
                                    ;7165           RC[TO]_Q,                       ;QUAD/DOUBLE.  PUT FIRST WORD IN TO
                                    ;7166           D_0,                            ;CLEAR SECOND WORD
U 0002, 0F01,203C,0180,F980,0000,0065 ;7167         J/A.I                           ;AND GO ON TO B FORK
                                    ;7168
                                    ;7169   003:    ;-------------------------------;
U 0003, 0000,003C,0180,F800,0000,0001 ;7170         J/RSVMOD                        ;RESERVED MODE
                                    ;7171
                                    ;7172   004:    ;-------------------------------;
U 0004, F000,003F,01F0,F847,0000,0200 ;7173         B.FORK                          ;REGISTER.  GO RIGHT ON TO B FORK
                                    ;7174
                                    ;7175   005:    ;-------------------------------;
U 0005, 0C00,003C,0180,F800,0000,0001 ;7176         J/RSVMOD
                                    ;7177
                                    ;7178   006:    ;-------------------------------;
U 0006, 0001,003C,0180,F980,0000,0011 ;7179         RC[TO]_D                        ;GET LOW-ADDR WORD TO TO
                                    ;7180
                                    ;7181           ;-------------------------------;
U 0011, 0800,003C,0180,F860,0000,0065 ;7182         D_R(PRN+1),J/A.I                ;GET SECOND PART
                                    ;7183
                                    ;7184   007:    ;-------------------------------;
U 0007, 0000,003C,0180,F800,0000,0001 ;7185         J/RSVMOD                        ;
```

```
                                    ;7186   ;A-FORK SPECIFIER EVALUATION, CONTINUED
                                    ;7187
                                    ;7188   008:    ;-------------------------------;
                                    ;7189   A.D1:   VA_D,                          ;(R) MODE.
U 0008, 0001,083C,0180,F800,0200,0094 ;7190         DATA.TYPE?,J/A.M
                                    ;7191
                                    ;7192   009:    ;-------------------------------;
                                    ;7193           R(PRN)_D+K[SP1.CON].RLOG,      ;(R)+ AUTO INCREMENT
U 0009, 0019,0818,1580,F8D8,0000,0094 ;7194         DATA.TYPE?,J/A.M               ; USE UN-INCREMENTED ADDR IN D&VA
                                    ;7195
                                    ;7196   00A:    ;-------------------------------;
                                    ;7197           R(PRN)_D-K[SP1.CON].RLOG,      ;-(R) AUTO DECREMENT
                                    ;7198           D&VA_ALU,                      ; USE DECREMENTED ADDR
U 000A, 0819,0804,1580,F8D8,0200,0094 ;7199         DATA.TYPE?,J/A.M
                                    ;7200
                                    ;7201   00B:    ;-------------------------------;
                                    ;7202           R(PRN)_D+K[.4].RLOG,           ;@(R)+ AUTO INCREMENT DEFERED
                                    ;7203           D[LONG]_CACHE,
U 000B, 0019,0018,1180,40D8,0000,0008 ;7204         J/A.D1
                                    ;7205
                                    ;7206   ; ***********************************************
                                    ;7207   ; * Patch no. 050, PCS 000B trapped to WCS 1178 *
                                    ;7208   ; ***********************************************
                                    ;7209
                                    ;7210   00C:    ;-------------------------------;
                                    ;7211           RC[T7]_D.CTX,                  ;INDEX MODE, CONTEXT SHIFT INDEX
U 000C, 0061,C03D,0180,F9B8,0000,047E ;7212         CALL,J7ASPC                    ; AND GO EVALUATE BASE OPERAND ADDRESS
                                    ;7213
                                    ;7214   06C:    ;-------------------------------;
                                    ;7215           D&VA_D+LC,                     ;COMPUTE INDEXED BOA
U 006C, 0811,0814,0180,F800,0200,0094 ;7216         DATA.TYPE?,J/A.M               ;AND FINALLY GET THE OPERAND
                                    ;7217
                                    ;7218   00D:    ;-------------------------------;
                                    ;7219           D&VA_Q+LB.PC,                  ;D(R) DISPLACEMENT MODE.
                                    ;7220           CLR.IB.SPEC,                   ;DISCARD THE SPECIFIER
U 000D, D805,2814,0180,F800,0200,0094 ;7221         DATA.TYPE?,J/A.M               ;GO GET THE OPERAND
                                    ;7222
                                    ;7223   00F:    ;-------------------------------;
                                    ;7224           VA_Q+LB.PC,                    ;@D(R) DISPLACEMENT DEFERED
U 000F, D005,2014,0180,F800,0200,0022 ;7225         CLR.IB.SPEC                    ;DROP THE SPECIFIER
                                    ;7226
                                    ;7227   ; ***********************************************
                                    ;7228   ; * Patch no. 051, PCS 000F trapped to WCS 1179 *
                                    ;7229   ; ***********************************************
                                    ;7230
                                    ;7231           ;-------------------------------;
U 0022, 0000,003C,0180,4000,0000,0008 ;7232         D[LONG]_CACHE,J/A.D1           ;GET INDIRECT, GO USE IT AS ADDR
```

```
                                        ;7233   ;HERE ARE VARIANTS OF THE A-FORK ENTRY POINTS FOR R=PC
                                        ;7234
                                        ;7235   014:    ;--------------------------------;
U 0014, 0000,003C,0180,F800,0000,0001   ;7236           J/RSVMOD                        ;PC REGISTER MODE
                                        ;7237
                                        ;7238   015:    ;--------------------------------;
U 0015, 0000,003C,0180,F800,0000,0001   ;7239           J/RSVMOD                        ;ILLEGAL REGISTER MODE, R=PC
                                        ;7240
                                        ;7241   016:    ;--------------------------------;
U 0016, 0000,003C,0180,F800,0000,0001   ;7242           J/RSVMOD                        ;PC QUAD REGISTER MODE
                                        ;7243
                                        ;7244   017:    ;--------------------------------;
U 0017, 0000,003C,0180,F800,0000,0001   ;7245           J/RSVMOD                        ;ILLEGAL QUAD REGISTER MODE, R=PC
                                        ;7246
                                        ;7247   018:    ;--------------------------------;
U 0018, 0000,003C,0180,F800,0000,0001   ;7248           J/RSVMOD                        ;(PC)
                                        ;7249
                                        ;7250   019:    ;--------------------------------;
                                        ;7251           D_Q,                            ;(PC)+ IMMEDIATE MODE
                                        ;7252           Q_PC,                           ;GET PC IN CASE ASRC
                                        ;7253           CLR.IB.SPEC,                    ;IRD CLEARED THE LITERAL, GET THE SPEC
U 0019, DC14,0838,01C0,F800,0000,0065   ;7254           DATA.TYPE?,J/A.I                ; LOOK OUT FOR ADDRESS SOURCE
                                        ;7255
                                        ;7256   01A:    ;--------------------------------;
U 001A, 0000,003C,0180,F800,0000,0001   ;7257           J/RSVMOD                        ;-(PC)
                                        ;7258
                                        ;7259   01B:    ;--------------------------------;
                                        ;7260           VA_Q,D_Q,                       ;@(PC)+ ABSOLUTE MODE
                                        ;7261           CLR.IB.SPEC,
U 001B, DC01,283C,0180,F800,0200,0094   ;7262           DATA.TYPE?,J/A.M
                                        ;7263
                                        ;7264   01C:    ;--------------------------------;
U 001C, 0000,003C,0180,F800,0000,0001   ;7265           J/RSVMOD                        ;INDEX MODE, R=PC
                                        ;7266
                                        ;7267   01D:    ;--------------------------------;
U 001D, 0000,003C,0180,F800,0000,0001   ;7268           J/RSVMOD                        ;NESTED INDEX MODE, R=PC
                                        ;7269
                                        ;7270   01F:    ;--------------------------------;
                                        ;7271           RC[T0]_Q,                       ;QUAD IMMEDIATE
                                        ;7272           Q_IB.DATA,                      ;GET SECOND LONGWORD OF LITERAL
                                        ;7273           CLR.IB.COND,                    ;THROW IT AWAY
                                        ;7274           PC_PC+4,                        ;ADVANCE PC
U 001F, F001,283C,01F0,F986,0000,0090   ;7275           IB.TEST?,J/A.IQ                 ;MAKE SURE IT'S ALL THERE
```

```
                                    ;7276   ;HERE OFF A-FORK WHEN THERE IS A TRAP, INTERRUPT, OR CONSOLE REQUEST UP
                                    ;7277
                                    ;7278   ;020:    ;--------------------------------;
                                    ;7279   ;       J/020                           ;NO SUCH CONDITION
                                    ;7280
                                    ;7281   ;024:    ;--------------------------------;SEE TP FAULT ROUTINE
                                    ;7282   ;       J/TRACE                         ;TRACE TRAP PENDING
                                    ;7283
                                    ;7284   ;028:    ;--------------------------------;NO SUCH CONDITION
                                    ;7285   ;       J/028
                                    ;7286
                                    ;7287   ;02C:    ;--------------------------------;NO SUCH CONDITION
                                    ;7288   ;       J/02C
                                    ;7289
                                    ;7290   030:
                                    ;7291   FO.ABS.30:
                                    ;7292           ;--------------------------------;
U 0030, 0000,003C,0180,F800,0000,0F8F ;7293         J/INTRPT                        ;INTERNAL INTERRUPT REQUEST
                                    ;7294
                                    ;7295   034:
                                    ;7296   FO.ABS.34:
                                    ;7297           ;--------------------------------;HALT PENDING
U 0034, 0814,0038,0180,F800,0000,005A ;7298         D_PC                            ;UNDO EXTRA PC ADVANCE
                                    ;7299
                                    ;7300           ;--------------------------------;
                                    ;7301           PC&VA_D-K[.1],                  ;SET PC TO NEXT INSTRUCTION TO DO
U 005A, 0019,0000,0580,F801,0200,0439 ;7302         J/CONS.HALT                     ;GO TO HALT LOOP
                                    ;7303
                                    ;7304   038:
                                    ;7305   FO.ABS.38:
                                    ;7306           ;--------------------------------;
U 0038, 0000,003C,0180,F800,0000,0F8D ;7307         J/INTIO                         ;SBI INTERRUPT REQUEST
                                    ;7308
                                    ;7309   ;03C:    ;--------------------------------;SEE ARITHMETIC TRAP ROUTINE
                                    ;7310   ;       J/ARITHT                        ;ARITHMETIC TRAP
                                    ;7311
                                    ;7312
                                    ;7313   ;HERE OFF A-FORK IF INSTRUCTION BUFFER DOES NOT HAVE ENOUGH DATA
                                    ;7314
                                    ;7315   060:
                                    ;7316   FO.ABS.60:
                                    ;7317           ;00--------------------------------;
U 0060, 0000,003D,0180,F800,0000,0E64 ;7318         CALL,J/IB.TBM                   ;STOPPED WITH TB MISS.  GO FILL TB
                                    ;7319
                                    ;7320   061:
                                    ;7321   FO.ABS.61:
                                    ;7322           ;01--------------------------------;
U 0061, 0000,003D,0180,F800,0000,0B80 ;7323         CALL,J/IB.ERR                   ;STOPPED WITH AN ERROR
                                    ;7324
                                    ;7325   062:    ;10--------------------------------;
U 0062, F80C,003B,01F1,F857,1398,6000 ;7326 IRD:    IRD                             ;WAITING FOR DATA.  LOOP ON IT
```

```
                                        ;7327   ;HERE FOR THE SECOND AND SUBSEQUENT STATES OFF A-FORK
                                        ;7328
                                        ;7329   =100    ;00--------------------------------;GET HERE BY DATA.TYPE?
                                        ;7330   A.M:    D_CACHE.INST.DEP,                   ;NORMAL B, W, L, OR F DATA
U 0094, F000,C03F,01F0,5847,0000,0200   ;7331           B.FORK                              ;GO EVALUATE SECOND SPECIFIER
                                        ;7332
                                        ;7333           ;01--------------------------------;
                                        ;7334           D_CACHE.INST.DEP,                   ;QUAD OR DOUBLE, GET TWO LONGWORDS
U 0095, 0000,C03C,0180,5800,0000,0064   ;7335           J7A.MQ
                                        ;7336                                               ;FIELD SOURCE DOESN'T OCCUR ON A-FORK
                                        ;7337   =111    ;11--------------------------------;
U 0097, F000,003F,01F0,F847,0000,0200   ;7338           B.FORK                              ;ADDRESS SOURCE IS ALREADY IN D
                                        ;7339
                                        ;7340           ;--------------------------------;
                                        ;7341   A.MQ:    RC[TO]_D,                           ;SAVE FIRST PART OF QUAD/DOUBLE OP
                                        ;7342           ID_D.SYNC,                          ;THROW IT ONTO BUS FOR ACCEL TO GRAB
U 0064, 0001,007C,1580,BD83,0000,0073   ;7343           VA_VA+4                             ; AND ADDRESS SECOND PART
                                        ;7344
                                        ;7345           ;--------------------------------;
                                        ;7346           D[LONG]_CACHE,                      ;GET SECOND PART
U 0073, F000,003F,01F0,4047,0000,0200   ;7347           B.FORK                              ;GO TO NEXT SPECIFIER
                                        ;7348
                                        ;7349   ;HERE FOR QUAD/DOUBLE I-STREAM LITERALS
                                        ;7350   ;THE FIRST LONGWORD OF LITERAL IS IN TO ALREADY, WE'VE TRIED TO READ THE
                                        ;7351   ; SECOND LONGWORD, AND HERE WE TEST TO SEE IF WE GOT IT.
                                        ;7352
                                        ;7353   =00     ;00--------------------------------;
U 0090, 0000,003D,0180,F800,0000,0E64   ;7354   A.IQ:   CALL,J/IB.TBM                        ;ISTREAM HAD A TB MISS
                                        ;7355
                                        ;7356           ;01--------------------------------;
U 0091, 0000,003D,0180,F800,0000,0B80   ;7357           CALL,J/IB.ERR                       ;I BUFFER STOPPED FOR AN ERROR
                                        ;7358
                                        ;7359           ;10--------------------------------;IB IS WAITING FOR DATA TO ARRIVE
                                        ;7360           Q_IB.DATA,                          ;STALL WAITING FOR THE DATA
                                        ;7361           C[R.IB2-5,
U 0092, F000,0B3C,01F0,F800,0000,0090   ;7362           IB.TEST?,J/A.IQ                     ;LOOP UNTIL IT ARRIVES
                                        ;7363
                                        ;7364           ;11--------------------------------;
                                        ;7365           D_Q,                                ;GOT IT
                                        ;7366           Q_PC,                               ;GET PC IN CASE ASRC
                                        ;7367           C[R.IB.SPEC,                        ;CLR SPECIFIER
U 0093, DC14,0838,01C0,F800,0000,0065   ;7368           DATA.TYPE?                          ;BEWARE ADDRESS SOURCES
                                        ;7369
                                        ;7370   =101    ;--------------------------------;ASRC+VSRC =0
U 0065, F000,003F,01F0,F847,0000,0200   ;7371   A.I:    B.FORK                              ;NORMAL, OPERAND IS SET UP
                                        ;7372
                                        ;7373           ;--------------------------------;ASRC+VSRC =1
                                        ;7374           D_D-K[SP1.CON],                     ;GET BACK TO UN-INCREMENTED PC
U 0067, F819,0C03,15F0,F847,0000,0200   ;7375           B.FORK
                                        ;7376
                                        ;7377   ; ************************************************
                                        ;7378   ; * Patch no. 045, PCS 0067 trapped to WCS 116D *
                                        ;7379   ; ************************************************
```

```
                                        ;7380   ;A-FORK OPTIMIZATIONS: SHORT LITERAL-REGISTER, AND REGISTER-REGISTER
                                        ;7381
                                        ;7382   ;HERE FOR CERTAIN INTEGER INSTRUCTIONS WHICH DO NOT WRITE A DESTINATION,
                                        ;7383   ; NAMELY BITB, BITW, BITL, CMPB, CMPW, CMPL
                                        ;7384   ; WITH THE FIRST SPECIFIER SHORT LITERAL, AND THE SECOND REGISTER MODE.
                                        ;7385
                                        ;7386   080:    ;-------------------------------;
                                        ;7387           ALU_LA[INST.DEP]Q,              ;OPERATE REGISTER AGAINST LITERAL
                                        ;7388           SET.CC(INST),                   ; SET THE CONDITION CODES IN PSL
                                        ;7389           CLR.IB0-1,                      ;STEP OVER OPCODE & SRC2
                                        ;7390           PC_PC+2,                        ;KEEP PC IN SYNC
U 0080, 401C,C00C,0180,F805,4070,0062   ;7391           J/IRD                           ;GO TO NEXT INSTR
                                        ;7392
                                        ;7393   ;HERE FOR INTEGER INSTRUCTIONS WHICH DO NOT WRITE A DESTINATION,
                                        ;7394   ; NAMELY BITB, BITW, BITL, CMPB, CMPW, CMPL
                                        ;7395   ; WITH BOTH SPECIFIERS INDICATING REGISTER MODE OPERANDS.
                                        ;7396
                                        ;7397   084:    ;-------------------------------;
                                        ;7398           ALU_LA[INST.DEP]LB,             ;PERFORM COMPARE-TYPE OPERATION
                                        ;7399           SET.CC(INST),                   ;SET CONDITION CODES IN PSL
                                        ;7400           CLR.IB0-1,                      ;DROP OPC & SRC2 SPEC
                                        ;7401           PC_PC+2,
U 0084, 400C,C00C,0180,F805,4070,0062   ;7402           J/IRD
                                        ;7403
                                        ;7404   ;HERE FOR CERTAIN INTEGER AND BOOLE INSTRUCTIONS WHICH WRITE A DESTINATION,
                                        ;7405   ; NAMELY ADDx2, SUBx2, BISx2, BICx2, XORx2, MNEGx, MOVx, MCOMx
                                        ;7406   ; FOR x=B, W, L, AND ADWC, SBWC
                                        ;7407   ; WITH THE FIRST SPECIFIER SHORT LITERAL, AND THE SECOND REGISTER MODE.
                                        ;7408
                                        ;7409   0C0:    ;-------------------------------;
                                        ;7410           ALU_LA[INST.DEP]Q,              ;SL-R OPERATION
                                        ;7411           R(SP1)_ALU,                     ;RESULT INTO DST REGISTER
                                        ;7412           SET.CC(INST),                   ;SET CONDITION CODES ACCORDINGLY
                                        ;7413           CLR.IB0-1,                      ;DROP OPCODE & DST SPEC
                                        ;7414           PC_PC+2,
U 00C0, 401C,C00C,0180,F8C5,4070,0062   ;7415           J/IRD
                                        ;7416
                                        ;7417   ;HERE FOR CERTAIN INTEGER AND BOOLE INSTRUCTIONS WHICH WRITE A DESTINATION,
                                        ;7418   ; NAMELY ADDx2, SUBx2, BISx2, BICx2, XORx2, MNEGx, MOVx, MCOMx
                                        ;7419   ; FOR x=B, W, L, AND ADWC, SBWC
                                        ;7420   ; WITH BOTH SPECIFIERS INDICATING REGISTER MODE OPERANDS.
                                        ;7421
                                        ;7422   0C4:    ;-------------------------------;
                                        ;7423           ALU_LA[INST.DEP]LB,             ;R-R OPERATION
                                        ;7424           R(SP1)_ALU,                     ;RESULT TO DEST REGISTER
                                        ;7425           SET.CC(INST),                   ;CC ARE INSTR DEPENDENT
                                        ;7426           CLR.IB0-1,                      ;GO TO NEXT INSTR
U 00C4, 400C,C00C,0180,F8C5,4070,0062   ;7427           PC_PC+2,J/IRD                   ;
```

```
                                        ;7428   ;HERE FOR CERTAIN SINGLE-OPERAND INTEGER INSTRUCTIONS,
                                        ;7429   ; NAMELY INCx, DECx FOR x=B, W, L
                                        ;7430   ; WITH THE SPECIFIER IN REGISTER MODE.
                                        ;7431
                                        ;7432   02F:    ;-------------------------------;(S-CLASS)
                                        ;7433   INC.R:
                                        ;7434   DEC.R:  R(PRN) D[INST.DEP]Q,           ;OPERATE ON THE REGISTER
                                        ;7435           SET.CC(INST),                   ;PSL CC SET ACCORDINGLY
                                        ;7436           CLR.IB.OPC,                     ;GO TO NEXT INSTR
                                        ;7437           PC_PC+1,                        ;BUMP PC ACCORDINGLY
U 002F, C01D,C00C,0180,F8DC,4070,0062   ;7438           J/IRD
                                        ;7439
                                        ;7440   ;HERE FOR SOBGTR, SOBGEQ WITH DESTINATION REGISTER
                                        ;7441
                                        ;7442   02E:    ;-------------------------------;
                                        ;7443   SOB.R:  R(PRN) D-K[.1].RLOG,           ;DECREMENT REGISTER
                                        ;7444           SET.CC(LONG),                   ;PSL CONDITION CODES GIVE BR CONDITION
                                        ;7445           Q_IB.BDEST,                     ;GET BDEST
                                        ;7446           PC_PC+1,                        ;STEP PC PAST IT
U 002E, 7019,0B04,05F0,F8DC,0070,00E0   ;7447           IB.TEST?
                                        ;7448
                                        ;7449   =00     ;00-------------------------------;
U 00E0, 0000,003D,0180,F800,0000,0E64   ;7450   SOB.R1: CALL,J/IB.TBM
                                        ;7451
                                        ;7452           ;01-------------------------------;
U 00E1, 0000,003D,0180,F800,0000,0B80   ;7453           CALL,J/IB.ERR
                                        ;7454
                                        ;7455           ;10-------------------------------;
                                        ;7456           Q_IB.BDEST,                     ;WAIT FOR IB TO GET BDEST
U 00E2, 7000,0B3C,01F0,F800,0000,00E0   ;7457           IB.TEST?,J/SOB.R1               ;LOOP ON IT
                                        ;7458
                                        ;7459           ;11-------------------------------;
                                        ;7460           Q_Q+PC,                         ;COMPUTE BRANCH DESTINATION ADDR
                                        ;7461           CLR.IB.SPEC,                    ;DISCARD BDEST FROM BYTE 1
U 00E3, D015,3A14,01C0,F800,0000,0163   ;7462           PSL.CC?,J/SOB.2                 ;DECIDE WHETHER TO BRANCH
                                        ;7463
                                        ;7464   ;HERE FOR BLBS, BLBC IN REGISTER MODE
                                        ;7465
                                        ;7466   0AE:    ;-------------------------------;
                                        ;7467   BLB.R:  ALU_D[INST.DEP]K[.1],           ;TEST LOW BIT OF REGISTER
                                        ;7468           CLK.UBCC,                       ;SET ALU Z FROM IT
                                        ;7469           Q_IB.BDEST,                     ;GET BRANCH DESTINATION FROM IB
                                        ;7470           PC_PC+1,                        ;STEP PC OVER IT
U 00AE, 7019,0B0C,05F0,F804,0010,0388   ;7471           IB.TEST?,J/BLB.1                ;WAIT FOR IT IF NECESSARY
                                        ;7472
                                        ;7473   ;HERE FOR TSTB, TSTW, TSTL IN REGISTER MODE
                                        ;7474
                                        ;7475   0AF:    ;-------------------------------;
                                        ;7476   TST.R:  ALU_D,SET.CC(INST),            ;SET CC ON REGISTER
U 00AF, C001,C03C,0180,F804,4070,0062   ;7477           CLR.IB.OPC,PC_PC+1,J/IRD        ;GO TO NEXT INSTRUCTION
                                        ;7478
                                        ;7479   0A9:    ;-------------------------------;
                                        ;7480   TSTF.R: ALU_D,SET.CC(INST),            ;TEST FLOAT/DOUBLE ON REGISTER
                                        ;7481           CHK.FLT.OPR,                    ;FAULT IF RESERVED OPERAND
U 00A9, C001,C03C,0180,F804,4870,0062   ;7482           CLR.IB.OPC,PC_PC+1,J/IRD        ;GO TO NEXT INSTRUCTION
```

```
                                        ;7483   ;HERE ARE EXECUTION STATES OFF A-FORK
                                        ;7484
                                        ;7485   ;HERE FOR BSBB, BSBW
                                        ;7486   ;BRANCH DISPLACEMENT IS IN Q, SIGN EXTENDED BY THE INSTRUCTION BUFFER
                                        ;7487
                                        ;7488   082:    ;-------------------------------;
                                        ;7489   BSB:    LAB_R[SP],                      ;SETUP SP FOR PUSHING PC
U 0082, 0814,0038,0180,FA70,0000,0076   ;7490           D_PC                            ;GET PC READY IN D
                                        ;7491
                                        ;7492           ;-------------------------------;
U 0076, 0018,0004,1180,FAF0,0200,00A6   ;7493           R[SP]&VA_LA-K[.4].RLOG          ;PUSH SP
                                        ;7494
                                        ;7495           ;-------------------------------;
U 00A6, 0000,003C,0180,3000,0000,00C8   ;7496           CACHE_D[LONG]                   ;STORE THE OLD PC
                                        ;7497                                           ; AND FALL INTO SUCCESSFUL BRANCH
                                        ;7498
                                        ;7499   ;HERE FOR "SUCCESSFUL" BRANCHES (THOSE WHICH CHANGE PC)
                                        ;7500   ;BRANCH DISPLACEMENT IS IN Q, SIGN EXTENDED BY THE INSTRUCTION BUFFER
                                        ;7501
                                        ;7502   0C8:    ;-------------------------------;
                                        ;7503           PC&VA_Q+PC,
                                        ;7504           FLUSH.IB,
U 00C8, 2015,2014,0180,F801,4200,00AB   ;7505           J/IB.FILL
                                        ;7506
                                        ;7507   0CD:    ;-------------------------------;
                                        ;7508           PC&VA_Q+PC,
                                        ;7509           FLUSH.IB,
U 00CD, 2015,2014,0180,F801,4200,00AB   ;7510           J/IB.FILL
                                        ;7511
                                        ;7512   0CE:    ;-------------------------------;
                                        ;7513   BR:     PC&VA_Q+PC,                     ;ADD DISPLACEMENT TO PC
U 00CE, 2015,2014,0180,F801,4200,00AB   ;7514           FLUSH.IB
                                        ;7515
                                        ;7516           ;-------------------------------;
                                        ;7517   IB.FILL:PC_PC+1,                        ;SKIP PC PAST OPCODE OF NEW INSTR
U 00AB, 0000,003C,0180,6004,0000,0062   ;7518           LOAD.IB,J/IRD                   ;FILL IB WITH NEW INSTR
                                        ;7519
                                        ;7520   ;HERE FOR "FAILURE" BRANCHES (NOP'S)
                                        ;7521
                                        ;7522   0C9:    ;-------------------------------;
                                        ;7523           CLR.IB.OPC,
U 00C9, C000,003C,0180,F804,4000,0062   ;7524           PC_PC+1,J/IRD
                                        ;7525
                                        ;7526   0CA:    ;-------------------------------;
                                        ;7527           CLR.IB.OPC,
U 00CA, C000,003C,0180,F804,4000,0062   ;7528           PC_PC+1,J/IRD
                                        ;7529
                                        ;7530   0CC:    ;-------------------------------;
                                        ;7531           CLR.IB.OPC,                     ;BRANCH FAIL, GO TO NEXT IN LINE
U 00CC, C000,003C,0180,F804,4000,0062   ;7532           PC_PC+1,J/IRD                   ;
```

```
E 10  PCS microcode patches : Patches prior to WCS116
F 10  PCS microcode patches : Patches prior to WCS116
G 10  PCS microcode patches : Patches prior to WCS116
H 10  PCS microcode patches : Patches prior to WCS116
I 10  PCS microcode patches : Patches prior to WCS116
J 10  PCS microcode patches : Patches prior to WCS116
K 10  PCS microcode patches : Patches prior to WCS116
L 10  PCS microcode patches : Patches prior to WCS116
M 10  PCS microcode patches : Patches prior to WCS116
N 10  PCS microcode patches : Patches prior to WCS116
B 11  PCS microcode patches : Patches first included in WCS116
C 11  PCS microcode patches : Patches first included in WCS116
D 11  PCS microcode patches : Patches first included in WCS117
E 11  PCS microcode patches : Patches first included in WCS117
F 11  PCS microcode patches : Patches first included in WCS117
G 11  PCS microcode patches : Patches first included in WCS117
H 11  PCS microcode patches : Patches first included in WCS117
I 11  PCS microcode patches : Patches first included in WCS117
J 11  PCS microcode patches : Patches first included in WCS117
K 11  PCS microcode patches : Patches first included in WCS117
L 11  PCS microcode patches : Patches first included in WCS117
M 11  PCS microcode patches : Patches first included in WCS117
N 11  PCS microcode patches : Patches first included in WCS117
B 12  PCS microcode patches : Patches first included in WCS117
C 12  PCS microcode patches : Patches first included in WCS117
D 12  PCS microcode patches : Patches first included in WCS117
E 12  PCS microcode patches : Patches first included in WCS117
F 12  PCS microcode patches : Patches first included in WCS117
G 12  PCS microcode patches : Patches first included in WCS120
H 12  PCS microcode patches : Patches first included in WCS120
I 12  PCS microcode patches : Patches first included in WCS120
J 12  PCS microcode patches : Patches first included in WCS120
K 12  PCS microcode patches : Patches first included in WCS120
L 12  PCS microcode patches : Patches first included in WCS120
M 12  PCS microcode patches : Patches first included in WCS120
N 12  PCS microcode patches : Patches first included in WCS120
B 13  PCS microcode patches : Patches first included in WCS120
C 13  PCS microcode patches : Patches first included in WCS120
D 13  PCS microcode patches : Patches first included in WCS120
E 13  PCS microcode patches : Patches first included in WCS120
F 13  PCS microcode patches : Patches first included in WCS120
G 13  PCS microcode patches : Patches first included in WCS120
H 13  PCS microcode patches : Patches first included in WCS120
I 13  PCS microcode patches : Patches first included in WCS120
J 13  PCS microcode patches : Patches first included in WCS120
K 13  PCS microcode patches : Patches first included in WCS120
L 13  PCS microcode patches : Patches first included in WCS120
M 13  PCS microcode patches : Patches first included in WCS120
N 13  PCS microcode patches : Patches first included in WCS122
B 14  PCS microcode patches : Patches first included in WCS123
C 14  PCS microcode patches : Patches first included in WCS123
D 14  PCS microcode patches : Patches first included in WCS123
E 14  PCS microcode patches : Patches first included in WCS123
F 14  PCS microcode patches : Patches first included in WCS123
G 14  PCS microcode patches : Patches first included in WCS123
H 14  PCS microcode patches : Patches first included in WCS123
I 14  PCS microcode patches : Patches first included in WCS123
J 14  PCS microcode patches : Patches first included in WCS123
K 14  PCS microcode patches : Patches first included in WCS123
L 14  PCS microcode patches : Patches first included in WCS123
M 14  PCS microcode patches : Patches first included in WCS123
N 14  PCS microcode patches : Patches first included in WCS123
B 15  PCS microcode patches : Patches first included in WCS123
C 15  PCS microcode patches : Patches first included in WCS123
D 15  PCS microcode patches : Patches first included in WCS123
E 15  PCS microcode patches : Patches first included in WCS123
F 15  PCS microcode patches : Patches first included in WCS123
G 15  PCS microcode patches : Patches first included in WCS123
H 15  PCS microcode patches : Patches first included in WCS123
I 15  PCS microcode patches : Patches first included in WCS123
J 15  PCS microcode patches : Patches first included in WCS123
K 15  PCS microcode patches : Patches first included in WCS123
L 15  PCS microcode patches : Patches first included in WCS123
M 15  PCS microcode patches : Patches first included in WCS123
N 15  PCS microcode patches : Patches first included in WCS123
B 16  PCS microcode patches : Patches first included in WCS123
C 16  PCS microcode patches : Patches first included in WCS123
D 16  FORKS.MIC
E 16  I-stream decode forks : A-FORK for VAX Instructions
F 16  I-stream decode forks : A-FORK for VAX Instructions
G 16  I-stream decode forks : A-FORK for VAX Instructions
H 16  I-stream decode forks : A-FORK for VAX Instructions
I 16  I-stream decode forks : A-FORK for VAX Instructions
J 16  I-stream decode forks : A-FORK for VAX Instructions
K 16  I-stream decode forks : A-FORK for VAX Instructions
L 16  I-stream decode forks : A-FORK for VAX Instructions
```