

Table of contents

4-	1	DBGENT -- Enter debugger to start program
5-	1	DBGTRP -- Entry from trap
6-	1	DBGBRK -- Enter debugger by typing ctrl-D
7-	1	BRKENT -- Enter debugger from breakpoint
8-	1	EXIT -- Exit from debugger to user
9-	1	NEWCMD -- Get new command
10-	1	-- Command processing routines --
10-	2	"/" -- Examine word
10-	17	"\" -- Examine byte
11-	1	"[" -- Decode value as an instruction
12-	1	"@" -- Open indirect
12-	28	" " -- Open cell indirect, relative to PC
13-	1	"!" -- Convert address to relocation offset
14-	1	"R" -- Set relocation base register
15-	1	"B" -- Set breakpoint
16-	1	"G" -- Start program execution
16-	12	"P" -- Proceed from breakpoint
16-	22	"S" -- Set/Reset single-step mode
16-	34	"I" -- Set/Reset I-space value inspection
17-	1	"TAB" -- Single step the program
18-	1	"X" -- Display RAD50 value
19-	1	"M" -- Monitor data cell
20-	1	"CR" -- Close current location
20-	13	"LF" -- Close location and advance to next
21-	1	"]" -- Close location and advance to instruction
22-	1	"^" -- Close current location and go to previous
23-	1	-- Subroutines --
23-	2	DBGINI -- Initialize the debugger
24-	1	BRKRST -- Reset breakpoint instructions
25-	1	BRKSET -- Set breakpoints
26-	1	BRKCHK -- Determine type of breakpoint
27-	1	SHOBRK -- Display breakpoint information
28-	1	FLGBRK -- Set flag if any breakpoints are active
29-	1	ACRVAL -- Accrue a general value
30-	1	ACRNUM -- Accrue a number
31-	1	GETCHR -- Get next character from terminal
31-	34	PSHCHR -- Push a character
32-	1	PRTOCT -- Print octal value
33-	1	PRTWRD -- Print octal word value
33-	24	PRTBYT -- Print octal byte value
34-	1	PRTR50 -- Print a RAD50 value
35-	1	ERRPRT -- Print an error message
35-	12	LSTTXT -- Print text string
36-	1	SETLOC -- Set address of open cell
37-	1	STRVAL -- Store value into open cell
38-	1	GETVAL -- Get value from current cell
39-	1	SHOLOC -- Display current address
40-	1	LSTADR -- Display address in user's program
41-	1	SHOADR -- Display address in user's program
42-	1	RELADR -- Determine if address is in relocation region
43-	1	DOPRT -- Print a text string
44-	1	DBGON -- Place terminal in debug mode
44-	9	DBGOFF -- Turn off terminal debug mode
44-	17	CHKADR -- Determine if address is valid
45-	1	DECODE -- Decode instruction into symbolic form
46-	1	ODCxxx -- Display instruction operands
56-	1	ODCGEN -- Display general operand value

TSDEBUG -- TSX-Plus debugging mo MACRO V05.05 Wednesday 18-Jan-89 15:24

Table of contents

60-	1	ODCREG -- Display register name
61-	1	ODCACC -- Print a floating point accumulator name
62-	1	ODCFPU -- Print general floating point operand
63-	1	ODCADR -- Display decoded instruction address
64-	1	ODCNEG -- Determine if values should be shown as negative
65-	1	ODCTAB -- Tab to a specified column
66-	1	INSBAS -- Instruction decoding tables

```

1          .TITLE  TSDBUG -- TSX-Plus debugging module
2          .ENABL  LC
3          .DSABL  GBL
4          .ENABL  AMA
5 000000    .CSECT  TSDBUG
6 000000    014527  TSDBUG: .RAD50  /DBG/          ;System overlay id
7          ;-----
8          ; This module implements the TSX-Plus program debugging facility.
9          ;
10         ; Copyright (c) 1984.
11         ; S&H Computer Systems, Inc.
12         ; Nashville, Tennessee USA
13         ; All rights reserved.
14         ;
15         ; Macro calls
16         ;
17         .MCALL  .TTYIN, .TTYOUT
18         ;
19         ; Global definitions
20         ;
21         .GLOBL  DBGENT, BRKENT, DBGBRK, DBGTRP
22         ;
23         ; Global references
24         ;
25         .GLOBL  D.BYTM, D.BKAD, D.RLBS, D.SVCH, D.NMBF, D.LOC
26         .GLOBL  D.VAL1, D.V1FL, D.VAL2, D.V2FL, D.LOCM
27         .GLOBL  D.R0, D.R1, D.R2, D.R3, D.R4, D.R5, D.R6, D.R7
28         .GLOBL  D.FLAG, D$SSTP, D$DBRK, D$IBRK, D$SBRK, D.BKNM
29         .GLOBL  D.BKSV, D$DMON, D.DADR, D.DOLD, D.DTRG
30         .GLOBL  D$DVAL, D$IPND, D$BKST, D.PS, UMODE, UPMODE
31         .GLOBL  D.SPSV, D.NMBE, D.START, D.END, D$FBRK
32         .GLOBL  PUTCHR, CORUSR, OVRHC, UPMODE, PSW, D.PCNT
33         .GLOBL  LSW9, $DBG$BK, D$RUN, D.LVAL, D.MASK, D$CKBK
34         .GLOBL  D$INIT, $DEBUG, D.PCOL, D.ILEN, D$TSTP
35         .GLOBL  D.PFMT, DP$DAA, DP$LAA, D.CBRK
36         .GLOBL  D$ISPC
37         ;
38         ; Macro definitions
39         ;
40         .MACRO  OCALL  ENTADD
41         CALL    OVRHC
42         .WORD   ENTADD
43         .ENDM   OCALL
44         ;
45         .MACRO  ERR      ERRMSG
46         MOV     ERRMSG, R1
47         JMP     ERRPRT
48         .ENDM   ERR
49         ;
50         .MACRO  TPRINT  STRING
51         .IF     NB STRING
52         MOV     STRING, R0
53         .ENDC
54         CALL    DOPRT
55         .ENDM   TPRINT
56         ;
57         .MACRO  PRINT  STRING

```

```

58          JSR      R2,LSTTXT
59          .ASCIZ   "'STRING'"
60          .EVEN
61          .ENDM    PRINT
62          ;
63          ; EMT argument blocks to control terminal debug mode
64          ;
65 000002      001      111      ONEMT: .BYTE 1,111          ;Turn on debug mode
66 000004      000      111      OFFEMT: .BYTE 0,111        ;Turn off debug mode
67          ;
68          ; Ascii character values
69          ;
70          CR       =      15          ; Carriage-return
71          LF       =      12          ; Line-feed
72          BELL     =      07          ; Bell
73          BKSPAC   =      10          ; Backspace
74          TAB      =      11          ; Tab
75          SPACE    =      40          ; Space
76          COMMA    =      54          ; Comma
77          ;
78          ; Assembly parameters
79          ;
80          TRCTRP   =      20          ; Trace trap flag in PSW
81          COLOPN   =      8          ; Column number where operand is to go
82          ;
83          ; Error messages
84          ;
85          .NLIST   BEX
86 000006      111      156      166  EM$IRC: .ASCIZ /Invalid RAD50 character/
87 000036      111      156      166  EM$IVC: .ASCIZ /Invalid command/
88 000056      111      156      166  EM$IVL: .ASCIZ /Invalid value/
89 000074      111      156      166  EM$IIV: .ASCIZ /Invalid register/
90 000115      111      156      166  EM$IRB: .ASCIZ /Invalid relocation register/
91 000151      116      165      155  EM$NTL: .ASCIZ /Number too long/
92 000171      015      012      124  TM$GRT: .ASCIZ <CR><LF>/TSX-Plus debugger/
93 000215      000
94 000216      015      200      CRCHAR: .BYTE CR,200        ;Print CR only
95          ;
96          ; RAD50 character table
97          ;
98 000220      040      101      102  R50CHR: .ASCII / ABCDEFGHIJKLMNOPQRSTUVWXYZ$. *0123456789/
99          .EVEN
100         .LIST    BEX

```

```

1
2      ; -----
3      ; Table of command characters
4      ;
5      CMDTBL: . BYTE      '/'      ; /
6              . BYTE      '\      ; \
7              . BYTE      '@      ; @
8              . BYTE      '_'      ; _
9              . BYTE      '!'      ; !
10             . BYTE      '['      ; [
11             . BYTE      ']'      ; ]
12             . BYTE      'B      ; B
13             . BYTE      'G      ; G
14             . BYTE      'I      ; I
15             . BYTE      'M      ; M
16             . BYTE      'P      ; P
17             . BYTE      'R      ; R
18             . BYTE      'S      ; S
19             . BYTE      'X      ; X
20             . BYTE      CR      ; Carriage-return
21             . BYTE      LF      ; Line-feed
22             . BYTE      '^      ; Up arrow
23             . BYTE      BKSPAC  ; Backspace
24             . BYTE      TAB      ; Tab
25
26      NUMCMD = .-CMDTBL
27             . EVEN
28
29      ; Parallel table of command processing routines
30      ;
31      CMDVEC: . WORD      CMDSLH      ; /
32              . WORD      CMDBKS      ; \
33              . WORD      CMDAT      ; @
34              . WORD      CMDUS      ; _
35              . WORD      CMDEXP      ; !
36              . WORD      CMDDCD      ; [
37              . WORD      CMDRSB      ; ]
38              . WORD      CMDB      ; B
39              . WORD      CMDG      ; G
40              . WORD      CMDI      ; I
41              . WORD      CDM      ; M
42              . WORD      CMDP      ; P
43              . WORD      CMDR      ; R
44              . WORD      CMDS      ; S
45              . WORD      CMDX      ; X
46              . WORD      CMDCR      ; Carriage-return
47              . WORD      CMDLF      ; Line-feed
48              . WORD      CMDUP      ; Up arrow
49              . WORD      CMDBSP      ; Backspace
50              . WORD      CMDTAB      ; Tab

```

```

1
2
3
4 000364      060
5 000365      061
6 000366      062
7 000367      063
8 000370      064
9 000371      065
10 000372     066
11 000373     067
12 000374     123
13 000375     115
14 000376     106
15 000377     102
16 000400     122
17           000015
18
19
20
21
22 000402     000000G
23 000404     000000G
24 000406     000000G
25 000410     000000G
26 000412     000000G
27 000414     000000G
28 000416     000000G
29 000420     000000G
30 000422     000000G
31 000424     000000G
32 000426     000000G
33 000430     000000G
34 000432     000000G
35 000434

;-----
; Table of internal register names
;
INTCHR: .BYTE '0          ; $0
        .BYTE '1          ; $1
        .BYTE '2          ; $2
        .BYTE '3          ; $3
        .BYTE '4          ; $4
        .BYTE '5          ; $5
        .BYTE '6          ; $6 (SP)
        .BYTE '7          ; $7 (PC)
        .BYTE 'S          ; $S (PS)
        .BYTE 'M          ; $M (Mask word)
        .BYTE 'F          ; $F (Format register)
        .BYTE 'B          ; $B (Breakpoint locations)
        .BYTE 'R          ; $R (Relocation base addresses)
NUMINT  = .-INTCHR
        .EVEN
;
; Parallel table of addresses of internal register cells
;
INTADR: .WORD D.R0         ; $0
        .WORD D.R1         ; $1
        .WORD D.R2         ; $2
        .WORD D.R3         ; $3
        .WORD D.R4         ; $4
        .WORD D.R5         ; $5
        .WORD D.R6         ; $6 (SP)
        .WORD D.R7         ; $7 (PC)
        .WORD D.PS         ; $S (PS)
        .WORD D.MASK       ; $M (Mask word)
        .WORD D.PFMT       ; $F (Format register)
        .WORD D.BKAD       ; $B (Breakpoint locations)
        .WORD D.RLBS       ; $R (Relocation base addresses)
INTEND:
; End of table of addresses

```

1				.SBTTL DBGENT -- Enter debugger to start program
2				-----
3				; DBGENT is jumped to when a program is started under the debugger.
4				; The following information is on the stack:
5				; 0(SP) = PC of start point.
6				; 2(SP) = PS to start with.
7				;
8	000434			DBGENT:
9				;
10				; Initialize the debugger
11				;
12	000434	004737	004226'	CALL DBGINI ; Initialize the debugger
13				;
14				; Set starting address and initial PSW
15				;
16	000440	012637	000000G	MOV (SP)+,D.R7 ; Set starting address
17	000444	012637	000000G	MOV (SP)+,D.PS ; Set starting PSW
18				;
19				; Enter debugger
20				;
21	000450	000137	001054'	JMP DBGRUN ; Enter debugger

```

1          .SBTTL  DBGTRP -- Entry from trap
2          ;-----
3          ; DBGTRP is jumped to if a trap occurs in a program running under the
4          ; debugger.
5          ; On entry, the following values are set up:
6          ;   0(SP) = Saved R5
7          ;   2(SP) = Saved R4
8          ;   4(SP) = Trap PC
9          ;   6(SP) = Trap PS
10         ;   R5    = Trap code (1==>Trap 4, 2==>Trap 10)
11         ;   R4    = Non-zero ==> Trap within debugger
12         ;
13 000454   DBGTRP:
14         ;
15         ; See if the trap occurred in the debugger or in the user's program
16         ;
17 000454   005704       TST      R4          ;Trap in debugger or user's program?
18 000456   001077       BNE      10$        ;Br if trap occurred in the debugger
19         ;
20         ; Trap occurred in the user's program.
21         ; Save entry information.
22         ;
23 000460   012637   000000G       MOV      (SP)+, D, R5
24 000464   012637   000000G       MOV      (SP)+, D, R4
25 000470   012637   000000G       MOV      (SP)+, D, R7          ;Trap PC
26 000474   012637   000000G       MOV      (SP)+, D, PS          ;Trap PSW
27 000500   010037   000000G       MOV      R0, D, R0
28         ;
29         ; Print trap entry message
30         ;
31 000504       TPRINT   #CRLF          ;Go to a new line
32 000514   020527   000001       CMP      R5, #1          ;Trap to 4 or 10?
33 000520   001017       BNE      1$          ;Br if trap to 10
34 000522       PRINT   <Trap to 4 at location >
35 000556   000416       BR       2$
36 000560   1$: PRINT   <Trap to 10 at location >
37 000614   013700   000000G   2$: MOV      D, R7, R0          ;Get address where trap occurred
38 000620   004737   007276'   CALL      LSTADR          ;Display the address
39 000624   4$: TPRINT   #CRLF          ;Go to new line
40         ;
41         ; Enter the debugger
42         ;
43 000634   042737   000000C 000000G   BIC      #<D$DBRK!D$IBRK!D$SBRK>, D, FLAG ;Say no breakpoint occurred
44 000642   013704   000000G       MOV      D, R4, R4          ;Restore R4 and R5
45 000646   013705   000000G       MOV      D, R5, R5
46 000652   000137   001054'       JMP      DBGRUN          ;Enter the debugger
47         ;
48         ; A trap occurred within the debugger.
49         ; Restore context to condition at the time of the trap and then
50         ; reenter the debugger.
51         ;
52 000656   013706   000000G   10$: MOV      D, SPSV, SP          ;Restore the stack pointer
53 000662   052737   000000G 000000G   BIS      #UPMODE, @#PSW          ;Make sure previous mode = user
54 000670   005037   000000G       CLR      D, LOC          ;No currently open cell
55 000674   000137   001510'       JMP      NEWCMD          ;Reenter the debugger

```

DBGBRK -- Enter debugger by typing ctrl-D

```

1          SBTTL  DBGBRK -- Enter debugger by typing ctrl-D
2          ;-----
3          ;  DBGBRK is entered when the user requests entry to the debugger by typing
4          ;  ctrl-D.
5          ;
6          ;  On entry, the stack is set up as follows:
7          ;    0(SP) = PC of next instruction.
8          ;    2(SP) = PS of next instruction.
9          ;
10         DBGBRK:
11         ;
12         ;  Reset job control flag that was used to force the breakpoint
13         ;
14         000700 010146          MOV      R1,-(SP)
15         000702 113701 000000G  MOVB     CORUSR,R1          ;Get job index number
16         000706 042761 000000G 000000G  BIC      #$DBGBK,LSW9(R1);Clear flag saying to force debugger break
17         000714 052761 000000G 000000G  BIS      #$DEBUG,LSW9(R1);Say program is now running with debugger
18         000722 012601          MOV      (SP)+,R1
19         ;
20         ;  If we have not initialized the debugger, do the debugger startup
21         ;
22         000724 032737 000000G 000000G  BIT      #D$INIT,D.FLAG  ;Has initialization been done?
23         000732 001002          BNE      1$          ;Br if yes
24         000734 004737 004226'        CALL     DBGINI          ;Initialize the debugger
25         ;
26         ;  Enter debugger as if we hit a breakpoint
27         ;
28         000740 052737 000000G 000000G 1$:  BIS      #D$FBRK,D.FLAG  ;Set flag saying we had a forced break
29         000746 000137 000752'        JMP      BRKENT          ;Now enter debugger as if we had a breakpoint

```

```

1          .SBTTL  BRKENT -- Enter debugger from breakpoint
2          ;-----
3          ; BRKENT is jumped to when a breakpoint trap occurs.
4          ; The following information is on the stack:
5          ;   0(SP) = PC of breakpoint trap.
6          ;   2(SP) = PS of breakpoint trap.
7          ;
8 000752 011637 000000G BRKENT: MOV      (SP),D.R7      ;Save breakpoint PC
9 000756 010037 000000G      MOV      R0,D.R0      ;Save user's R0
10         ;
11         ; Determine if we should enter the debugger
12         ;
13 000762 004737 004456'      CALL      BRKCHK      ;See if breakpoint causes program stop
14 000766 032737 000000C 000000G      BIT      #<D$DBRK!D$IBRK!D$SBRK!D$FBRK>,D.FLAG ;Breakpoint?
15 000774 001023          BNE      5$      ;Br if this break will enter debugger
16         ;
17         ; This breakpoint will not enter the debugger.
18         ; If we just executed an instruction that was under a breakpoint,
19         ; set breakpoints and then continue program execution.
20         ;
21 000776 042737 000000G 000000G      BIC      #D$IPND,D.FLAG ;Say pending instruction has been executed
22 001004 032737 000000C 000000G      BIT      #<D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Are we single stepping
23 001012 001006          BNE      3$      ;Br if yes
24 001014 004737 004376'      CALL      BRKSET      ;Set breakpoints
25 001020 042766 000020 000002      BIC      #TRCTRP,2(SP) ;Reset trace trap flag in PSW
26 001026 000403          BR      2$
27 001030 052766 000020 000002 3$:  BIS      #TRCTRP,2(SP) ;Set trap flag in PSW on stack
28 001036 013700 000000G      2$:  MOV      D.R0,R0      ;Recover R0
29 001042 000006          RTT      ;Return from breakpoint
30         ;
31         ; This breakpoint will enter the debugger.
32         ; Save information and enter the debugger
33         ;
34 001044 012637 000000G      5$:  MOV      (SP)+,D.R7      ;Save trap PC
35 001050 012637 000000G      MOV      (SP)+,D.PS      ;Save trap PS
36         ;
37         ; Save user's registers
38         ;
39 001054 052737 000000G 000000G DBGRUN: BIS      #D$RUN,D.FLAG ;Set flag saying debugger is running
40 001062 010137 000000G      MOV      R1,D.R1
41 001066 010237 000000G      MOV      R2,D.R2
42 001072 010337 000000G      MOV      R3,D.R3
43 001076 010437 000000G      MOV      R4,D.R4
44 001102 010537 000000G      MOV      R5,D.R5
45 001106 106506          MFPD      SP
46 001110 012637 000000G      MOV      (SP)+,D.R6
47         ;
48         ; Save initial stack pointer in case we need to restore it if an
49         ; error occurs.
50         ;
51 001114 010637 000000G      MOV      SP,D.SPSV      ;Save initial stack pointer
52         ;
53         ; Clear the trace-trap flag in the user's TRAP instruction PSW.
54         ; We may have set the trace-trap flag in this PSW if we were single
55         ; stepping the program.
56         ;
57 001120 106537 000036          MFPD      @#36      ;Get user's TRAP PSW

```

```

58 001124 042716 000020          BIC    #TRCTRP,(SP)    ;Clear the trace-trap flag
59 001130 106637 000036          MTPD    @#36          ;Store new TRAP PSW
60
61                               ;
62                               ; If breakpoints are set in program, restore the real instructions
63                               ; that belong in the breakpoint locations.
64 001134 042737 000000G 000000G    BIC    #D$IPND,D.FLAG    ;Clear instruction-pending flag
65 001142 032737 000000C 000000G    BIT    #<D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Are we single stepping?
66 001150 001014                BNE     2$            ;Br if yes
67 001152 004737 004302'          CALL    BRKRST          ;Replace breakpoint instructions
68 001156 032737 000000G 000000G    BIT    #D$IBRK,D.FLAG    ;Did an instruction breakpoint occur?
69 001164 001406                BEQ     2$            ;Br if not
70 001166 162737 000002 000000G    SUB     #2,D.R7        ;Backup PC to point to BPT instruction
71 001174 052737 000000G 000000G    BIS     #D$IPND,D.FLAG    ;Remember instruction must be executed
72
73                               ; See if a proceed repeat count is in effect
74                               ;
75 001202 005737 000000G          2$:    TST     D.PCNT        ;Is a proceed repeat count in effect?
76 001206 001407                BEQ     3$            ;Br if not
77 001210 032737 000000G 000000G    BIT     #D$FBRK,D.FLAG    ;Is this a forced breakpoint?
78 001216 001003                BNE     3$            ;Br if yes -- always stop
79 001220 005337 000000G          DEC     D.PCNT        ;Should we skip this breakpoint?
80 001224 001011                BNE     EXIT1          ;Br if yes
81
82                               ; Clear temporary single-step flag
83                               ;
84 001226 042737 000000G 000000G 3$:    BIC     #D$TSTP,D.FLAG    ;Clear temporary single-step flag
85
86                               ; Put terminal control in debug mode
87                               ;
88 001234 004737 007614'          CALL     DBGON          ;Put terminal in debug mode
89
90                               ; Display information about the breakpoint
91                               ;
92 001240 004737 004670'          CALL     SHOBRK          ;Display breakpoint information
93 001244 000137 001510'          JMP      NEWCMD          ;Enter the debugger

```

EXIT -- Exit from debugger to user

```

1          .SBTTL  EXIT  -- Exit from debugger to user
2          ;-----
3          ; Exit from debugger to user's program.
4          ;
5 001250 042737 000000G 000000G EXIT1: BIC      #D$FBRK,D.FLAG ;Turn off forced-breakpoint flag
6 001256 032737 000000C 000000G      BIT      #<D$SSTP!D$TSTP!D$DMON>,D.FLAG ;Are we single stepping?
7 001264 001403                      BEQ      3$          ;Br if not
8 001266 042737 000000G 000000G      BIC      #D$IPND,D.FLAG ;Turn of instruction-pending if single step
9 001274 005037 000000G          3$:  CLR      D.LOC      ;Say no location is open
10 001300 042737 000000G 000000G      BIC      #D$RUN,D.FLAG ;Say debugger is no longer running
11 001306 004737 007624'          CALL     DBGOFF      ;Turn off terminal debug mode
12 001312 032737 000000C 000000G      BIT      #<D$IPND!D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Should we single step?
13 001320 001003                      BNE      4$          ;Br if yes
14 001322 004737 004376'          CALL     BRKSET      ;Set breakpoints
15 001326 000406                      BR       1$
16          ;
17          ; We are single stepping the program.
18          ; Set the trace-trap flag in the user's TRAP instruction PSW so we
19          ; will continue to have program control during processing of TRAP
20          ; instructions.
21          ;
22 001330 106537 000036          4$:  MFPD      @#36          ;Get user's TRAP PSW
23 001334 052716 000020          BIS      #TRCTRP,(SP) ;Set the trace-trap flag in the PSW
24 001340 106637 000036          MTPD      @#36          ;Store new TRAP PSW
25          ;
26          ; If we are doing data monitoring, save current value of monitored cell
27          ;
28 001344 032737 000000G 000000G 1$:  BIT      #D$DMON,D.FLAG ;Are we doing data cell monitoring?
29 001352 001411                      BEQ      2$          ;Br if not
30 001354 013705 000000G          MOV      D.DADR,R5      ;Get address of cell being monitored
31 001360 106515                      MFPD      (R5)      ;Get current value from cell
32 001362 013704 000000G          MOV      D.MASK,R4      ;Get current data mask
33 001366 005104                      COM      R4          ;Complement
34 001370 040405                      BIC      R4,R5      ;Clear all but bits of interest
35 001372 012637 000000G          MOV      (SP)+,D.DOLD ;Save old value
36          ;
37          ; Restore user's registers
38          ;
39 001376 013746 000000G          2$:  MOV      D.R6,-(SP)
40 001402 106606                      MTPD      SP
41 001404 013705 000000G          MOV      D.R5,R5
42 001410 013704 000000G          MOV      D.R4,R4
43 001414 013703 000000G          MOV      D.R3,R3
44 001420 013702 000000G          MOV      D.R2,R2
45 001424 013701 000000G          MOV      D.R1,R1
46          ;
47          ; Exit point used if we did not enter debugger
48          ;
49 001430 013700 000000G          EXIT2: MOV      D.R0,R0
50          ;
51          ; See if we need to set Trap flag in PS
52          ;
53 001434 042737 000020 000000G          BIC      #TRCTRP,D.PS ;Reset trace trap flag in PS
54 001442 032737 000000C 000000G          BIT      #<D$IPND!D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Do we want single step?
55 001450 001403                      BEQ      1$          ;Br if not
56 001452 052737 000020 000000G          BIS      #TRCTRP,D.PS ;Set Trap flag in user's PS
57 001460 052737 000000C 000000G 1$:  BIS      #UMODE!UPMODE,D.PS ;Make sure user-mode is set in PS

```

EXIT -- Exit from debugger to user

58

59

60

61 001466 013746 000000G

62 001472 013746 000000G

63 001476 000006

;

;

;

Use RTT instruction to reenter user's program

MOV D.PS, -(SP)

MOV D.R7, -(SP)

RTT

; Set PS for user

; Set PC for user

; Enter user's program

1				.SBTTL NEWCMD -- Get new command
2				-----
3				; Print carriage-return, line-feed and get new command
4				;
5	001500			NEWLIN: TPRINT #CRLF
6				;
7				; Get new command
8				;
9	001510			NEWCMD: PRINT <DBG:>
10				;
11				; Assume neither value1 nor value2 is specified
12				;
13	001522	105037	000000G	GETCMD: CLRB D.V1FL ;Value1 not specified
14	001526	005037	000000G	CLR D.VAL1
15	001532	105037	000000G	CLRB D.V2FL ;Value2 not specified
16	001536	005037	000000G	CLR D.VAL2
17				;
18				; See if Value1 is specified
19				;
20	001542	004737	005310'	CALL ACRVAL ;Try to accrue value1
21	001546	010037	000000G	MOV RO,D.VAL1 ;Save value for value1
22	001552	110137	000000G	MOVB R1,D.V1FL ;Remember if value1 specified
23				;
24				; See if value2 is specified
25				;
26	001556	004737	005756'	CALL GETCHR ;Get delimiter
27	001562	120027	000073	CMPB RO,'#'; ;is there a value2?
28	001566	001403		BEG 1\$;Br if possibly yes
29	001570	004737	006022'	CALL PSHCHR ;Save command character
30	001574	000406		BR 2\$
31	001576	004737	005310'	1\$: CALL ACRVAL ;Get value2
32	001602	010037	000000G	MOV RO,D.VAL2 ;Save value2
33	001606	110137	000000G	MOVB R1,D.V2FL ;Save info about value2
34				;
35				; Look up command character
36				;
37	001612	004737	005756'	2\$: CALL GETCHR ;Get command character
38	001616	012701	000024	MOV #NUMCMD,R1 ;Get # valid command characters
39	001622	120061	000270'	3\$: CMPB RO,CMDTBL(R1) ;Search for character in table
40	001626	001406		BEG 4\$;Br if found command character
41	001630	005301		DEC R1 ;More command chars to check?
42	001632	002373		BGE 3\$;Loop if yes
43	001634			ERR #EM\$IVC ;Invalid command character
44				;
45				; Found command character.
46				; Jump off to processing routine.
47				;
48	001644	006301		4\$: ASL R1 ;Convert command index into word table index
49	001646	000171	000314'	JMP @CMDVEC(R1) ;Jump to processing routine

-- Command processing routines --

```

1          .SBTTL  -- Command processing routines --
2          .SBTTL  "/"  -- Examine word
3          ; -----
4          ; "address/" -- Examine contents of word
5          ;
6 001652 004737 006426' CMDSLH: CALL SETLOC      ;Set value1 as current location
7 001656 032737 000001 000000G CMDSL1: BIT #1,D.LOC ;Is location odd?
8 001664 001021          BNE CMDBKS      ;If yes then treat "/" like "\"
9 001666 105037 000000G      CLR D.BYTM ;Say we are in word mode
10 001672 012737 000002 000000G      MOV #2,D.ILEN ;Say instruction length = 2 bytes
11 001700 004737 006714'      CALL GETVAL      ;Get contents of location
12 001704 010037 000000G      MOV R0,D.LVAL    ;Save last value displayed
13 001710 004737 006136'      CALL PRTWRD      ;Print the value
14 001714          PRINT < > ;Print two spaces
15 001724 000137 001522'      JMP GETCMD      ;Go get next command
16
17          .SBTTL  "\"  -- Examine byte
18          ; -----
19          ; "address\" -- Examine contents of a byte
20          ;
21 001730 004737 006426' CMDBKS: CALL SETLOC      ;Set value1 as current location
22 001734 112737 000001 000000G CMDBK1: MOVB #1,D.BYTM ;Say we are operating in byte mode
23 001742 112737 000001 000000G      MOVB #1,D.ILEN ;Say instruction length = 1 byte
24 001750 004737 006714'      CALL GETVAL      ;Get contents of location
25 001754 010037 000000G      MOV R0,D.LVAL    ;Save last value displayed
26 001760 010002          MOV R0,R2 ;Save value
27 001762 004737 006212'      CALL PRTBYT      ;Print value
28 001766          PRINT < = > ;Put in equal sign
29 001776 120227 000040      CMPB R2,#40 ;Is this a printing character?
30 002002 103002          BHIS 1$ ;Br if yes
31 002004 112702 000056      MOVB #'.,R2 ;Use period for non-printing chars
32 002010          1$: .TTYOUT R2 ;Print character
33 002016 005237 000000G      INC D.PCOL ;Advance print column
34 002022          PRINT < > ;Print 2 spaces
35 002032 000137 001522'      JMP GETCMD      ;Go get next command

```

"[" -- Decode value as an instruction

```

1          .SBTTL "[" -- Decode value as an instruction
2          ; -----
3          ;   Open a cell and display its contents as an instruction.
4          ;
5 002036   CMDDCD:
6          ;
7          ;   See if an address was specified
8          ;
9 002036   105737   000000G       TSTB      D.V1FL      ; Was an address specified?
10 002042   001402       BEQ        1$          ; Br if not
11 002044   004737   006426'     CALL      SETLOC     ; Set address of currently open cell
12 002050   013701   000000G     1$:      MOV      D.LOC,R1 ; Is a cell currently open?
13 002054   001002       BNE        2$          ; Br if yes
14 002056   000137   001500'     JMP        NEWLIN    ; Nothing to decode
15 002062   032701   000001     2$:      BIT        #1,R1 ; Is the address odd?
16 002066   001402       BEQ        3$          ; Br if not
17 002070   000137   001730'     JMP        CMDBKS    ; Treat like "\" if odd
18 002074   105737   000000G     3$:      TSTB      D.LOCM    ; Is cell internal or external?
19 002100   002002       BGE        4$          ; Br if external
20 002102   000137   001656'     JMP        CMDSL1    ; Treat [ like / for internal cell
21          ;
22          ;   Decode the instruction
23          ;
24 002106   004737   007640'     4$:      CALL      DECODE    ; Decode the instruction
25          ;
26          ;   Tab over some
27          ;
28 002112   013700   000000G       MOV      D.PCOL,R0    ; Get current print column
29 002116   062700   000010       ADD      #8,R0        ; Tab over some
30 002122   042700   000007       BIC      #7,R0
31 002126   004737   012064'     CALL      ODCTAB      ; Tab over
32          ;
33          ;   Finished
34          ;
35 002132   000137   001522'     JMP        GETCMD

```

"@" -- Open indirect

```

1          .SBTTL  "@" -- Open indirect
2          ;-----
3          ; Open the cell whose address is contained in the cell pointed to
4          ; by D.LOC.
5          ;
6 002136 013702 000000G CMDAT:  MOV    D.LVAL,R2      ;Get last displayed value
7 002142 010237 000000G CMDAT1: MOV    R2,D.LOC      ;Set as new location to display
8 002146 112737 000001 000000G      MOV    #1,D.LOCM    ;Say new address is in user's program
9 002154      TPRINT  #CRLF      ;Go to a new line
10         ;
11         ; See if we should do the display in byte or word mode
12         ;
13 002164 032702 000001      BIT     #1,R2      ;Should we display in byte or word mode?
14 002170 001010      BNE     1$      ;Br if need to go to byte mode
15         ;
16         ; Display new location in word mode
17         ;
18 002172 004737 007100'      CALL    SHOLOC      ;Display the current address
19 002176      PRINT    < / >      ;Display "/"
20 002206 000137 001656'      JMP     CMDSL1      ;Simulate the "/" command
21         ;
22         ; Display in byte mode
23         ;
24 002212 004737 007100' 1$: CALL    SHOLOC      ;Display the current address
25 002216      PRINT    < \ >      ;Display "\"
26 002226 000137 001734'      JMP     CMDBK1      ;Simulate the "\" command
27         ;
28         .SBTTL  "_" -- Open cell indirect, relative to PC
29         ;-----
30         ; "_" -- Open indirect relative to PC.
31         ;
32 002232 013702 000000G CMDUS:  MOV    D.LVAL,R2      ;Get last displayed value
33 002236 063702 000000G      ADD     D.LOC,R2      ;Add PC of instruction
34 002242 062702 000002      ADD     #2,R2      ;Plus 2
35 002246 000137 002142'      JMP     CMDAT1      ;Now treat like "@" command

```

"! " -- Convert address to relocation offset

```

1          .SBTTL  "! " -- Convert address to relocation offset
2          ;-----
3          ;  "n!" -- Convert current address to relocation offset
4          ;
5 002252    CMDEXP:
6          ;
7          ;  If no relocation register number was provided, find closest relocation
8          ;  base.
9          ;
10 002252   013701   000000G      MOV     D.VAL1,R1      ;Get relocation register #
11 002256   105737   000000G      TSTB    D.V1FL        ;Was a relocation register specified?
12 002262   003422                BLE     2$            ;Br if not
13          ;
14          ;  A relocation register number was provided.
15          ;  Display the relocation register number and the offset.
16          ;
17 002264                PRINT    < = >                ;Print "="
18 002274   010100                MOV     R1,R0          ;Get the relocation register number
19 002276   004737   006030'      CALL    PRTOCT        ;Print the relocation reg #
20 002302                PRINT    < , >                ;Print ", "
21 002310   006301                ASL     R1            ;Convert relocation # to word table index
22 002312   013700                MOV     D.LOC,R0      ;Get original value
23 002316   166100                SUB     D.RLBS(R1),R0 ;Subtract relocation base
24 002322   004737   006030'      CALL    PRTOCT        ;Print the offset
25 002326   000404                BR      8$
26          ;
27          ;  No relocation register number was specified
28          ;
29 002330   013700                2$:    MOV     D.LOC,R0      ;Get address
30 002334   004737   007322'      CALL    SHOADR        ;Display it with best relocation base
31 002340                8$:    PRINT    < >            ;Print 2 spaces
32          ;
33          ;  Finished
34          ;
35 002350   000137   001522'      9$:    JMP     GETCMD      ;Go get the next command

```

"R" -- Set relocation base register

```

1          .SBTTL  "R"  -- Set relocation base register
2          ;-----
3          ; "address;nR" -- Set relocation base register
4          ;
5 002354   CMDR:
6          ;
7          ; If no value was specified for argument 2, then we are being
8          ; asked to calculate the difference between the currently displayed
9          ; value and the base of the relocation area.
10         ; If argument 2 was specified, then we are setting the base of a
11         ; relocation area.
12         ;
13 002354   105737  000000G      TSTB    D.V2FL      ; Was argument 2 specified?
14 002360   001417          BEQ      2$            ; Br if not
15         ;
16         ; Set base address for relocation area
17         ;
18 002362   013701  000000G      MOV      D.VAL2,R1      ; Get register number
19 002366   020127  0000007      CMP      R1,#7          ; Should not exceed 7
20 002372   101404          BLOS     1$            ; Br if ok
21 002374          ERR      #EM$IVL
22 002404   006301          1$:    ASL      R1            ; Convert to word table index
23 002406   013761  000000G 000000G  MOV      D.VAL1,D.RLBS(R1); Set address for relocation base
24 002414   000137  001500'      JMP      NEWLIN        ; Go get next command
25         ;
26         ; Calculate offset relative to relocation base
27         ;
28 002420          2$:    PRINT    < = >          ; Print "="
29 002430   105737  000000G      TSTB    D.V1FL      ; Was a relocation reg # provided?
30 002434   001436          BEQ      4$            ; Br if not
31 002436   013701  000000G      MOV      D.VAL1,R1      ; Get register number
32 002442   020127  0000007      CMP      R1,#7          ; Should not exceed 7
33 002446   101404          BLOS     3$            ; Br if ok
34 002450          ERR      #EM$IVL
35 002460   006301          3$:    ASL      R1            ; Convert to word table index
36 002462   013700  000000G      MOV      D.LVAL,R0      ; Get currently open value
37 002466   166100  000000G      SUB      D.RLBS(R1),R0  ; Calculate offset
38 002472   010002          MOV      R0,R2            ; Hold the offset value
39 002474   010100          MOV      R1,R0            ; Get offset region #
40 002476   006200          ASR      R0            ; Convert to ordinal number
41 002500   004737  006030'      CALL     PRTOCT        ; Print relocation #
42 002504          PRINT    < , >          ; Print comma
43 002512   010200          MOV      R2,R0            ; Get offset value
44 002514   004737  006030'      CALL     PRTOCT        ; Print it
45 002520          PRINT    < >          ; Print a space
46 002530   000410          BR       9$
47         ;
48         ; No relocation register number was provided
49         ; Show address with best relocation
50         ;
51 002532   013700  000000G      4$:    MOV      D.LVAL,R0      ; Get value of interest
52 002536   004737  007322'      CALL     SHQADR        ; Display the address
53 002542          PRINT    < >          ; Print 2 spaces
54         ;
55         ; Finished
56         ;
57 002552   000137  001522'      9$:    JMP      GETCMD        ; Go get next command

```

"B" -- Set breakpoint

```

1          .SBTTL  "B"  -- Set breakpoint
2          ; -----
3          ;  "address;nB"  -- Set breakpoint
4          ;
5 002556 105737 000000G  CMDB:  TSTB  D.V2FL      ; Was a breakpoint number specified?
6 002562 001412          BEQ  1$          ; Br if not
7 002564 013702 000000G      MOV  D.VAL2,R2    ; Get breakpoint number
8 002570 006302          ASL  R2            ; Convert to word table index
9 002572 020227 000016      CMP  R2,#14.      ; Make sure it's not too big
10 002576 101416          BLOS 3$          ; Br if ok
11 002600          ERR  #EM$IVL
12 002610 005002 1$:      CLR  R2            ; Search for a free breakpoint entry
13 002612 005762 000000G  4$:      TST  D.BKAD(R2) ; Is this breakpoint free?
14 002616 001406          BEQ  3$          ; Br if yes
15 002620 062702 000002      ADD  #2,R2      ; Point to next breakpoint entry
16 002624 020227 000016      CMP  R2,#14.      ; Have we checked all breakpoint entries?
17 002630 101770          BLOS 4$          ; Br if more to check
18 002632 005002          CLR  R2            ; If all busy, then use # 0
19 002634 013701 000000G  3$:      MOV  D.VAL1,R1 ; Get address of breakpoint
20 002640 004737 007634'      CALL CHKADR    ; Make sure its ok
21 002644 010162 000000G      MOV  R1,D.BKAD(R2) ; Save address of breakpoint
22          ;
23          ; Set D$CKBK flag in D.FLAGS if there are any active breakpoints
24          ;
25 002650 004737 005244'      CALL  FLGBRK      ; Set flag if any breakpoints are active
26 002654 000137 001500'  9$:      JMP  NEWLIN    ; Go get next command

```

"G" -- Start program execution

```

1          .SBTTL "G" -- Start program execution
2          ;-----
3          ; "address;G" -- Start program execution.
4          ;
5 002660 105737 000000G CMDG: TSTB D.V1FL ;Was a starting address specified?
6 002664 003403 BLE 1$ ;Br if not
7 002666 013737 000000G 000000G MOV D.VAL1,D.R7 ;Set starting address
8 002674 1$: TPRINT #CRLF ;Go to new line
9 002704 005037 000000G CLR D.PCNT ;Clear any proceed count
10 002710 000137 001250' JMP EXIT1 ;Start running program
11
12          .SBTTL "P" -- Proceed from breakpoint
13          ;-----
14          ; "n;P" -- Proceed from breakpoint
15          ;
16 002714 105737 000000G CMDP: TSTB D.V1FL ;Was a proceed repeat count specified?
17 002720 003403 BLE 1$ ;Br if not
18 002722 013737 000000G 000000G MOV D.VAL1,D.PCNT ;Set proceed repeat count
19 002730 1$: TPRINT #CRLF ;Go to new line
20 002740 000137 001250' JMP EXIT1 ;Enter user's program
21
22          .SBTTL "S" -- Set/Reset single-step mode
23          ;-----
24          ; "n;S" -- Set or reset single-step mode
25          ;
26 002744 042737 000000G 000000G CMDS: BIC #D$SSTP,D.FLAG ;Assume he wants to reset single-step mode
27 002752 105737 000000G TSTB D.V2FL ;Was a value specified for "n"?
28 002756 001406 BEQ 1$ ;Br if not
29 002760 005737 000000G TST D.VAL2 ;Is value 2 non-zero?
30 002764 001403 BEQ 1$ ;If zero, don't single step
31 002766 052737 000000G 000000G BIS #D$SSTP,D.FLAG ;Set single-step mode
32 002774 000137 001500' 1$: JMP NEWLIN ;Go get next command
33
34          .SBTTL "I" -- Set/Reset I-space value inspection
35          ;-----
36          ; "n;I" -- Set or reset I-space value inspection (for / and \)
37          ; I-space is always used for instruction decoding
38          ;
39 003000 042737 000000G 000000G CMDI: BIC #D$ISPC,D.FLAG ;Assume / and \ refer to D-space
40 003006 105737 000000G TSTB D.V2FL ;Was a value specified for "n"?
41 003012 001406 BEQ 1$ ;Br if not
42 003014 005737 000000G TST D.VAL2 ;Was value zero?
43 003020 001403 BEQ 1$ ;0 or no value mean use D-space
44 003022 052737 000000G 000000G BIS #D$ISPC,D.FLAG ;Set I-space for / and \
45 003030 000137 001500' 1$: JMP NEWLIN ;Go get next command

```

"TAB"-- Single step the program

```

1          .SBTTL  "TAB"-- Single step the program
2          ;-----
3          ; The TAB command causes us to go into temporary single step mode
4          ; and to execute the next instruction.
5          ; If "valueTAB" is specified, we use the value as the address of
6          ; the next instruction to execute.
7          ; If "OTAB" is specified, we step over a subprogram call.
8          ;
9 003034    CMDTAB:
10         ;
11         ; See if a value was specified with TAB
12         ;
13 003034    105737    000000G      TSTB    D.V1FL      ;Was value 1 specified?
14 003040    001407      BEQ      2$      ;Br if not
15 003042    105037    000000G      CLRB    D.V1FL      ;Clear value-1 flag so it won't confuse ;P
16         ;
17         ; A value was specified with TAB.
18         ; If 0 was specified, skip over a subprogram call.
19         ; If a non-zero value was specified, use this as a starting address.
20         ;
21 003046    013700    000000G      1$:     MOV     D.VAL1,R0      ;Get specified value
22 003052    001407      BEQ      5$      ;0 ==> Skip over subprogram call
23         ;
24         ; A starting address was specified.
25         ;
26 003054    010037    000000G      MOV     R0,D.R7      ;Set starting address
27         ;
28         ; Single step the program.
29         ;
30 003060    052737    000000G 000000G 2$:     BIS     #D$TSTP,D.FLAG ;Say we are in temporary single step mode
31 003066    000137    002714'      JMP     CMDP      ;Now treat tab like ;P
32         ;
33         ; Argument value 1 is 0 (zero).
34         ; Step over a subprogram call.
35         ; Make sure the current instruction is a CALL.
36         ;
37 003072    013701    000000G      5$:     MOV     D.R7,R1      ;Get address of current instruction
38 003076    006521      MFPI      (R1)+      ;Get instruction from user's program
39 003100    012602      MOV     (SP)+,R2      ;Get the instruction from the stack
40 003102    010200      MOV     R2,R0      ;Copy the instruction word
41 003104    042700    000777      BIC     #777,R0      ;Clear register # and address field
42 003110    020027    004000      CMP     R0,#0004000    ;Is this a JSR instruction?
43 003114    001361      BNE     2$      ;If not, ignore the number in front of TAB
44         ;
45         ; Determine address of instruction beyond CALL and set it as a location
46         ; where a temporary breakpoint is to be set.
47         ;
48 003116    010200      MOV     R2,R0      ;Get the instruction
49 003120    042700    177770      BIC     #^C7,R0      ;Clear all but addressing register #
50 003124    020027    000007      CMP     R0,#7      ;Is it using register 7 (PC)?
51 003130    001003      BNE     3$      ;Br if not
52 003132    062701    000002      ADD     #2,R1      ;2 bytes for subroutine address if PC used
53 003136    000410      BR      4$
54 003140    010200      3$:     MOV     R2,R0      ;Get back the instruction
55 003142    042700    177707      BIC     #^C70,R0      ;Clear all but addressing mode
56 003146    072027    177775      ASH     #-3,R0      ;Right justify the addressing mode
57 003152    116000    003214'      MOVB   INSLN(R0),R0 ;Get # bytes used by this addressing mode

```

"TAB"-- Single step the program

```

58 003156 060001          ADD    R0,R1          ;Get address of location where breakpoint goes
59 003160 010137 000000G 4$:    MOV    R1,D.CBRK      ;Set address of call break point
60 003164 052737 000000G 000000G  BIS    #D$CKBK,D.FLAG  ;Set flag saying there are some breakpoints
61                                     ;
62                                     ; Reset single-step mode and continue with execution of the program
63                                     ;
64 003172 042737 000000G 000000G  BIC    #D$SSTP,D.FLAG  ;Reset single step mode
65 003200                                TPRINT  #CRLF      ;Go to a new line
66 003210 000137 001250'          JMP    EXIT1          ;Enter user's program
67                                     ;
68                                     ; Table providing number of bytes used by each addressing mode
69                                     ;
70 003214      000      000      000  INSLEN: .BYTE  0,0,0,0,0,0,2,2
    003217      000      000      000
    003222      002      002
71                                     . EVEN

```

"X" -- Display RAD50 value

```

1          .SBTTL "X" -- Display RAD50 value
2          ;-----
3          ; "X" -- Convert currently displayed value from RAD50 to ascii.
4          ;
5          CMDX: PRINT < = >          ;Print "="
6          003224 013700 000000G      MOV D.LVAL,R0          ;Get last displayed value
7          003234 004737 006270'      CALL PRTR50           ;Convert to ascii and print
8          003244          PRINT < >          ;Print spaces
9          003254 012737 000002 000000G  MOV #2,D.ILEN      ;Say instruction length = 2 bytes
10         ;
11         ; See if user wants to enter a new RAD50 value for this cell
12         ;
13         003262 004737 005756'      CALL GETCHR          ;Get 1st character
14         003266 004737 006022'      CALL PSHCHR          ;Push the character
15         003272 120027 000015      CMPB RO,#CR           ;No new value wanted?
16         003276 001454          BEQ 7$              ;Br if no new value specified
17         003300 120027 000012      CMPB RO,#LF           ;Advance to next cell?
18         003304 001451          BEQ 7$              ;Br if yes
19         ;
20         ; Now accept a new RAD50 value
21         ;
22         003306 005001          CLR R1                ;Form new RAD50 value in R1
23         003310 012702 000003      MOV #3,R2          ;Set count of # chars to accrue
24         003314 004737 005756'      1$: CALL GETCHR      ;Get next character of value
25         003320 120027 000015      CMPB RO,#CR          ;Reached end?
26         003324 001434          BEQ 2$              ;Br if yes
27         003326 120027 000012      CMPB RO,#LF          ;LF is also end
28         003332 001431          BEQ 2$
29         003334 012703 000220'      MOV #R50CHR,R3      ;Point to RAD50 character table
30         003340 120023      3$: CMPB RO,(R3)+          ;Search for char in table
31         003342 001407          BEQ 4$              ;Br if found
32         003344 020327 000270'      CMP R3,#R50CHR+50   ;Checked all of table?
33         003350 103773          BLO 3$              ;Loop if not
34         003352          ERR #EM$IRC                 ;Invalid RAD50 character
35         003362 162703 000221'      4$: SUB #R50CHR+1,R3 ;Get RAD50 character value
36         003366 070127 000050      MUL #50,R1          ;Multiply previous value by 50
37         003372 060301          ADD R3,R1             ;Add new character value
38         003374 077231          SOB R2,1$            ;Loop if more characters needed
39         003376 004737 005756'      6$: CALL GETCHR      ;Ignore other characters up to CR
40         003402 120027 000015      CMPB RO,#CR
41         003406 001403          BEQ 2$
42         003410 120027 000012      CMPB RO,#LF
43         003414 001370          BNE 6$
44         ;
45         ; We have gotten the RAD50 value. Store it in currently open cell
46         ;
47         003416 004737 006022'      2$: CALL PSHCHR      ;Save the terminating character
48         003422 010100          MOV R1,R0             ;Get value to R0 for STRVAL
49         003424 004737 006450'      CALL STRVAL        ;Store value into currently open cell
50         ;
51         ; If command ended with Line feed, open next cell
52         ;
53         003430 005037 000000G      7$: CLR D.PCOL      ;Say print column = 0
54         003434 004737 005756'      CALL GETCHR        ;Get the terminating character
55         003440 120027 000012      CMPB RO,#LF          ;Open the next cell?
56         003444 001004          BNE 9$              ;Br if not
57         003446 105037 000000G      CLRB D.V1FL        ;Tell LF not to store into current cell

```

"X" -- Display RAD50 value

```
58 003452 000137 003574'          JMP      CMDLF          ;Go open the next cell
59                                     ;
60                                     ; Finished
61                                     ;
62 003456 000137 001500'      9$:      JMP      NEWLIN
```

"M" -- Monitor data cell

```

1          .SBTTL  "M"  -- Monitor data cell
2          ; -----
3          ; "address;valueM" -- Monitor data cell
4          ;
5 003462  042737  000000C 000000G CMDM:  BIC      #<D$DMON!D$DVAL>,D.FLAG ;Reset data-cell monitoring flags
6 003470  005737  000000G          TST      D.VAL1          ;Was an address specified?
7 003474  001420          BEQ      9$          ;Br if not -- Disable monitoring
8 003476  052737  000000G 000000G          BIS      #D$DMON,D.FLAG ;Enable data monitoring
9 003504  013700  000000G          MOV      D.VAL1,RO        ;Get address of cell to monitor
10 003510  010037  000000G          MOV      RO,D.DADR        ;Set it as control address
11          ;
12          ; See if breakpoint is to be triggered whenever value in cell changes
13          ; or only when a specified value occurs.
14          ;
15 003514  105737  000000G          TSTB     D.V2FL          ;Was a value specified?
16 003520  003406          BLE      9$          ;Br if not
17 003522  052737  000000G 000000G          BIS      #D$DVAL,D.FLAG ;Watch for specified value
18 003530  013737  000000G 000000G          MOV      D.VAL2,D.DTRG ;Save the specified value
19          ;
20          ; Finished
21          ;
22 003536  000137  001500'          9$:      JMP      NEWLIN          ;Go get next command

```

"CR" -- Close current location

```

1          .SBTTL "CR" -- Close current location
2          ;-----
3          ; "carriage-return" -- Close current cell
4          ;
5 003542 005037 000000G CMDCR: CLR D.PCOL ; Say print column = 0
6 003546 105737 000000G      TSTB D.V1FL ; Was a new value specified for this cell?
7 003552 003404          BLE 9$ ; Br if not
8 003554 013700 000000G      MOV D.VAL1,RO ; Yes, get the value
9 003560 004737 006450'      CALL STRVAL ; Store value into current location
10 003564 005037 000000G 9$: CLR D.LOC ; Say no location open now
11 003570 000137 001510'      JMP NEWCMD ; Go get next command
12
13          .SBTTL "LF" -- Close location and advance to next
14          ;-----
15          ; "line-feed" -- Close current cell and advance to next
16          ;
17 003574 005037 000000G CMDLF: CLR D.PCOL ; Say print column = 0
18 003600 005737 000000G      TST D.LOC ; Is a location open now?
19 003604 001447          BEQ 5$ ; Br if not
20 003606 105737 000000G      TSTB D.V1FL ; Was a new value specified for this cell?
21 003612 003404          BLE 1$ ; Br if not
22 003614 013700 000000G      MOV D.VAL1,RO ; Get new value
23 003620 004737 006450'      CALL STRVAL ; Store into currently open cell
24 003624 105737 000000G 1$: TSTB D.BYTM ; Are we in byte or word mode?
25 003630 001017          BNE 2$ ; Br if in byte mode
26 003632 062737 000002 000000G ADD #2,D.LOC ; Advance location counter
27 003640          .TTYOUT #CR ; Go to new line
28 003650 004737 007100'      CALL SHOLDC ; Display address of new location
29 003654          PRINT < / > ; Display "/"
30 003664 000137 001656'      JMP CMDSL1 ; Simulate "/" command
31 003670 005237 000000G 2$: INC D.LOC ; Advance to next byte
32 003674          TPRINT #CRCHAR ; go to new line
33 003704 004737 007100'      CALL SHOLDC ; Display address of new location
34 003710          PRINT < \ > ; Display "\"
35 003720 000137 001734'      JMP CMDBK1 ; Simulate "\" command
36 003724 000137 001500' 5$: JMP NEWLIN ; No cell open -- Ignore LF

```

"J" -- Close location and advance to instruction

```

1          .SBTTL "J" -- Close location and advance to instruction
2          ;-----
3          ; "J" -- Close the current location and open next as an instruction.
4          ;
5 003730 005037 000000G CMDRSB: CLR      D.PCOL      ; Say print column = 0
6 003734 105737 000000G      TSTB     D.V1FL      ; Was an address specified?
7 003740 001402          BEQ      3$              ; Br if not
8 003742 000137 002036'      JMP      CMDDCD      ; Treat ] like [
9 003746 005737 000000G      3$: TST      D.LOC      ; Is a location open now?
10 003752 001440          BEQ      5$              ; Br if not
11 003754 105737 000000G      TSTB     D.BYTM      ; Are we in byte or word mode?
12 003760 001017          BNE      2$              ; Br if in byte mode
13 003762 063737 000000G 000000G      ADD      D.ILEN, D.LOC ; Advance location counter
14 003770          TPRINT    #CRLF      ; Go to new line
15 004000 004737 007100'      CALL     SHOLOC      ; Display address of new location
16 004004          PRINT     < [ >      ; Display "["
17 004014 000137 002036'      JMP      CMDDCD      ; Simulate "[" command
18 004020 005237 000000G      2$: INC      D.LOC      ; Advance to next byte
19 004024          TPRINT    #CRCHAR      ; go to new line
20 004034 004737 007100'      CALL     SHOLOC      ; Display address of new location
21 004040          PRINT     < \ >      ; Display "\"
22 004050 000137 001730'      JMP      CMDBKS      ; Simulate "\" command
23 004054 000137 001500'      5$: JMP      NEWLIN      ; No cell open -- Ignore LF

```

"^" -- Close current location and go to previous

```

1          .SBTTL  "^"  -- Close current location and go to previous
2          ; -----
3          ; "Backspace" -- Equivalent to up-arrow
4          ;
5 004060    CMDDBSP: .TTYOUT #136          ;Print "^"
6 004070    BR      CMDUP          ;Now treat backspace like up-arrow
7          ; -----
8          ;
9          ; "^" -- Close current location and go to previous one.
10         ;
11 004072    005037    000000G    CMDUP:  CLR      D.PCOL          ;Say print column = 0
12 004076    005737    000000G          TST      D.LOC          ;Is a cell open?
13 004102    001002          BNE      3$          ;Br if yes
14 004104    000137    001500'    JMP      NEWLIN          ;Ignore if no cell open
15 004110    105737    000000G    3$:  TSTB     D.V1FL          ;Was a new value specified for this cell?
16 004114    003404          BLE      1$          ;Br if not
17 004116    013700    000000G    MOV      D.VAL1,R0          ;Get new value
18 004122    004737    006450'    CALL     STRVAL          ;Store into currently open cell
19 004126    105737    000000G    1$:  TSTB     D.BYTM          ;Are we in byte or word mode?
20 004132    001017          BNE      2$          ;Br if in byte mode
21 004134    162737    000002    000000G    SUB      #2,D.LOC          ;Move to previous word
22 004142          TPRINT    #CRLF          ;Go to new line
23 004152    004737    007100'    CALL     SHOLOC          ;Display address of new location
24 004156          PRINT     < / >          ;Display "/"
25 004166    000137    001656'    JMP      CMDSL1          ;Simulate "/" command
26 004172    005337    000000G    2$:  DEC      D.LOC          ;Move to previous byte
27 004176          TPRINT    #CRLF          ;go to new line
28 004206    004737    007100'    CALL     SHOLOC          ;Display address of new location
29 004212          PRINT     < \ >          ;Display "\"
30 004222    000137    001734'    JMP      CMDBK1          ;Simulate "\" command

```

-- Subroutines --

```

1          .SBTTL  -- Subroutines --
2          .SBTTL  DBGINI -- Initialize the debugger
3          ; -----
4          ;   DBGINI is called to initialize the debugger.
5          ;   The debugger greeting message is also printed.
6          ;
7 004226 010046  DBGINI: MOV      RO, -(SP)
8          ;
9          ;   Initialize some debugger values
10         ;
11 004230 012700 000000G      MOV      #D.START, RO      ; Point to start of debugging data
12 004234 005020      1$:    CLR      (RO)+              ; Clear all debugging data to 0
13 004236 020027 000000G      CMP      RO, #D.END        ; Finished?
14 004242 103774      BLO      1$                        ; Loop if not
15 004244 012737 177777 000000G      MOV      #177777, D.MASK ; Initialize the data mask
16 004252 052737 000000G 000000G      BIS      #D$INIT, D.FLAG ; Say initialization has been done
17         ;
18         ;   Set previous-mode = user in our PSW so we can use MFPD/MTPD to
19         ;   access user's program.
20         ;
21 004260 052737 000000G 000000G      BIS      #UPMODE, @#PSW ; Say previous mode = user
22         ;
23         ;   Print greeting message
24         ;
25 004266      TPRINT  #TM$GRT          ; Print greeting message
26         ;
27         ;   Finished
28         ;
29 004276 012600      MOV      (SP)+, RO
30 004300 000207      RETURN

```

BRKRST -- Reset breakpoint instructions

```

1                                     .SBTTL  BRKRST -- Reset breakpoint instructions
2                                     ;-----
3                                     ; BRKRST is called on entry to the debugger when a breakpoint occurs.
4                                     ; It restores the contents of the instruction cells that were replaced
5                                     ; by BPT instructions for breakpoints.
6                                     ;
7 004302 010146 BRKRST: MOV      R1, -(SP)
8 004304 032737 000000G 000000G      BIT      #D$BKST, D.FLAG ;Are breakpoints set in program?
9 004312 001427      BEQ      9$      ;Br if not
10 004314 032737 000000G 000000G      BIT      #D$CKBK, D.FLAG ;Are there any active instruction breaks?
11 004322 001423      BEQ      9$      ;Br if not
12                                     ;
13                                     ; Reset instructions that were overwritten by breakpoints
14                                     ;
15 004324 005001      CLR      R1      ;Init index into breakpoint tables
16 004326 016100 000000G      1$:      MOV      D.BKAD(R1), R0 ;Get address of breakpoint
17 004332 001407      BEQ      2$      ;Br if this breakpoint not in use
18 004334 006510      MFPI      (R0)    ;Get breakpoint instruction
19 004336 022627 000003      CMP      (SP)+, #3 ;Is this a breakpoint instruction?
20 004342 001003      BNE      2$      ;Br if not
21 004344 016146 000000G      MOV      D.BKSV(R1), -(SP) ;Get saved instruction
22 004350 006610      MTPI      (R0)    ;Restore saved instruction
23 004352 062701 000002      2$:      ADD      #2, R1      ;Advance breakpoint index
24 004356 020127 000020      CMP      R1, #16. ;Checked all breakpoints?
25 004362 101761      BLOS      1$      ;Loop if more to check
26 004364 042737 000000G 000000G      BIC      #D$BKST, D.FLAG ;Say breakpoints no longer set
27                                     ;
28                                     ; Finished
29                                     ;
30 004372 012601      9$:      MOV      (SP)+, R1
31 004374 000207      RETURN

```

BRKSET -- Set breakpoints

```

1          .SBTTL  BRKSET -- Set breakpoints
2          ; -----
3          ; BRKSET is called when leaving the debugger and entering the user's program.
4          ; It sets BPT instructions in those locations marked for breakpoints.
5          ;
6 004376 010146 BRKSET: MOV      R1, -(SP)
7 004400 032737 000000G 000000G BIT      #D$CKBK, D. FLAG ;Are there any instruction breakpoints?
8 004406 001416 BEQ      3$          ;Br if not
9          ;
10         ; Set breakpoints
11         ;
12 004410 012701 000020 MOV      #16., R1          ;Get index for last breakpoint
13 004414 016100 000000G 1$: MOV      D. BKAD(R1), R0 ;Get address where breakpoint goes
14 004420 001406 BEQ      2$          ;Br if this breakpoint not in use
15 004422 006510 MFPI      (R0)          ;Get current contents of location
16 004424 012661 000000G MOV      (SP)+, D. BKSV(R1); Save original contents of cell
17 004430 012746 000003 MOV      #3, -(SP)          ;Push BPT instruction
18 004434 006610 MTPI      (R0)          ;Store BPT instruction into user's program
19 004436 162701 000002 2$: SUB      #2, R1          ;More breakpoints to check?
20 004442 002364 BGE      1$          ;Br if yes
21 004444 052737 000000G 000000G 3$: BIS      #D$BKST, D. FLAG ;Set flag saying breakpoints set in program
22         ;
23         ; Finished
24         ;
25 004452 012601 MOV      (SP)+, R1
26 004454 000207 RETURN

```

BRKCHK -- Determine type of breakpoint

```

1          .SBTTL  BRKCHK -- Determine type of breakpoint
2          -----
3          ; Determine the type of breakpoint that has occurred.
4          ; There are four different types of breakpoints:
5          ;   1. Data breakpoints that are triggered when a monitored cell changes
6          ;       value or takes on a specified value.
7          ;   2. Instruction breakpoints set by use of the "a;nB" instruction.
8          ;   3. Instruction breakpoint used to catch a CALL return.
9          ;   4. Single-step breakpoints that result from the use of the ";1S"
10         ;       command.
11         ;
12         ; Inputs:
13         ;   D.R7 = PC after the break
14         ;
15         ; Outputs:
16         ;   D$DBRK in D.FLAG ==> Data breakpoint
17         ;   D$IBRK in D.FLAG ==> Instruction breakpoint
18         ;   D$SBRK in D.FLAG ==> Single-step breakpoint or CALL breakpoint
19         ;   D$FBRK in D.FLAG ==> Forced breakpoint (user typed ctrl-B)
20         ;   D.BKNM = Breakpoint number
21         ;
22 004456 010146 BRKCHK: MOV      R1,-(SP)
23         ;
24         ; Clear instruction and data breakpoint flags
25         ;
26 004460 042737 000000C 000000G      BIC      #<D$DBRK!D$IBRK!D$SBRK>,D.FLAG ;No breakpoints yet
27         ;
28         ; See if a data breakpoint is active
29         ;
30 004466 032737 000000G 000000G      BIT      #D$DMON,D.FLAG ;Is a data breakpoint active?
31 004474 001425      BEQ      1$ ;Br if not
32 004476 013701 000000G      MOV      D.DADR,R1 ;Get address of monitored cell
33 004502 106511      MFPD     (R1) ;Get contents of monitored cell
34 004504 013701 000000G      MOV      D.MASK,R1 ;Get data mask
35 004510 005101      COM      R1 ;Complement
36 004512 040116      BIC      R1,(SP) ;Leave only bits of interest
37 004514 032737 000000G 000000G      BIT      #D$DVAL,D.FLAG ;Are we waiting for a specified value?
38 004522 001004      BNE      2$ ;Br if yes
39         ; Trigger data breakpoint whenever monitored data cell changes value
40 004524 022637 000000G      CMP      (SP)+,D.DOLD ;Has value changed?
41 004530 001004      BNE      3$ ;Br if yes -- trigger breakpoint
42 004532 000406      BR      1$ ;Hasn't changed -- no data break
43         ; Trigger data breakpoint if value takes on specified value
44 004534 022637 000000G 2$: CMP      (SP)+,D.DTRG ;Does cell have value of interest?
45 004540 001003      BNE      1$ ;Br if not -- no data breakpoint
46         ;
47         ; Trigger a data breakpoint
48         ;
49 004542 052737 000000G 000000G 3$: BIS      #D$DBRK,D.FLAG ;Set data-breakpoint flag
50         ;
51         ; See if we need to check for instruction or single-step breakpoints
52         ;
53 004550 032737 000000C 000000G 1$: BIT      #<D$SSTP!D$TSTP!D$CKBK>,D.FLAG ;Any instruction breaks or stp?
54 004556 001442      BEQ      9$ ;Br if neither instruction brks or single step
55         ;
56         ; Ignore single-step and instruction breakpoints if we just
57         ; executed a pending instruction.

```

BRKCHK -- Determine type of breakpoint

```

58
59 004560 032737 000000G 000000G ; BIT #D$IPND,D.FLAG ;Did we just execute a pending instruction?
60 004566 001036 ; BNE 9$ ;Br if yes
61
62 ; See if this is a single-step breakpoint
63
64 004570 032737 000000C 000000G ; BIT #D$SSTP!D$TSTP,D.FLAG ;Are we single stepping program?
65 004576 001407 ; BEQ 4$ ;Br if not
66 004600 052737 000000G 000000G 5$: BIS #D$SBRK,D.FLAG ;Set single-step breakpoint flag
67 004606 112737 000011 000000G MOVB #9.,D.BKNM ;Say this is breakpoint # 9
68 004614 000423 ; BR 9$
69
70 ; See if this is an instruction breakpoint
71
72 004616 012701 000020 4$: MOV #16.,R1 ;Init breakpoint index
73 004622 013700 000000G MOV D.R7,R0 ;Get address after breakpoint
74 004626 162700 000002 SUB #2,R0 ;Point to break instruction
75 004632 020061 000000G 7$: CMP R0,D.BKAD(R1) ;Search for which breakpoint was hit
76 004636 001404 ; BEQ 6$ ;Br if found it
77 004640 162701 000002 SUB #2,R1 ;Get next breakpoint index
78 004644 002372 ; BGE 7$ ;Loop if more breakpoint to check
79 004646 000406 ; BR 9$ ;This is not an instruction breakpoint
80
81 ; We hit an instruction breakpoint
82
83 004650 006201 6$: ASR R1 ;Get breakpoint number
84 004652 110137 000000G MOVB R1,D.BKNM
85 004656 052737 000000G 000000G BIS #D$IBRK,D.FLAG ;Remember we had an instruction break
86
87 ; Finished
88
89 004664 012601 9$: MOV (SP)+,R1
90 004666 000207 RETURN

```

SHOBRK -- Display breakpoint information

```

1          .SBTTL  SHOBRK -- Display breakpoint information
2          ;-----
3          ; SHOBRK is called to display information about a breakpoint
4          ; that has been triggered.
5          ;
6 004670 010146 SHOBRK: MOV      R1, -(SP)
7 004672          TPRINT  #CRLF          ;Get to start of new line
8          ;
9          ; See if a data breakpoint occurred
10         ;
11 004702 032737 000000G 000000G      BIT      #D$DBRK, D. FLAG ;Did a data breakpoint occur?
12 004710 001454          BEQ      1$          ;Br if not
13 004712          PRINT    <Data break at >
14 004736 013700 000000G      MOV      D. R7, RO          ;Get breakpoint PC
15 004742 004737 007276'      CALL    LSTADR          ;Display the address
16 004746          PRINT    < -- >
17 004760 013700 000000G      MOV      D. DADR, RO        ;Get address of monitored cell
18 004764 004737 007276'      CALL    LSTADR          ;Display it
19 004770          PRINT    < = >
20 005000 013700 000000G      MOV      D. DADR, RO        ;Get address of cell
21 005004 106510          MFPD      (RO)          ;Get current contents of cell
22 005006 012600          MOV      (SP)+, RO
23 005010 010037 000000G      MOV      RO, D. DOLD        ;Save old cell value
24 005014 004737 006136'      CALL    PRTWRD          ;Display value of cell
25 005020 013700 000000G      MOV      D. MASK, RO        ;Get data mask
26 005024 005100          COM      RO          ;Complement
27 005026 040037 000000G      BIC      RO, D. DOLD        ;Clear all but bits of interest
28 005032          TPRINT  #CRLF          ;Go to new line
29         ;
30         ; See if a single-step breakpoint occurred
31         ;
32 005042 032737 000000G 000000G 1$: BIT      #D$SBRK, D. FLAG ;Did a single-step breakpoint occur?
33 005050 001406          BEQ      2$          ;Br if not
34 005052          5$: PRINT    <Step:>          ;Print heading
35 005064 000451          BR       3$          ;Now treat like instruction breakpoing
36         ;
37         ; See if an instruction breakpoint occurred
38         ;
39 005066 032737 000000G 000000G 2$: BIT      #D$IBRK, D. FLAG ;Instruction breakpoint?
40 005074 001426          BEQ      4$          ;Br if not
41 005076          TPRINT  #CRLF          ;Go to new line
42 005106 123727 000000G 000010      CMPB    D. BKNM, #8.    ;CALL breakpoint?
43 005114 001003          BNE      6$          ;Br if not
44 005116 005037 000000G      CLR      D. CBRK          ;Clear the call breakpoint
45 005122 000753          BR       5$          ;Br if yes -- Treat like single step break
46 005124          6$: PRINT    <B>
47 005132 113700 000000G      MOVB     D. BKNM, RO        ;Get breakpoint number
48 005136 004737 006030'      CALL    PRT OCT          ;Print breakpoint number
49 005142          PRINT    < ; >
50 005150 000417          BR       3$
51         ;
52         ; See if we had a forced breakpoint
53         ;
54 005152 032737 000000G 000000G 4$: BIT      #D$FBRK, D. FLAG ;Forced breakpoint?
55 005160 001427          BEQ      7$          ;Br if not
56 005162          TPRINT  #CRLF          ;Go to a new line
57 005172          PRINT    <Break at >

```

SHOBRK -- Display breakpoint information

```

58      ;
59      ;   Display the address of the breakpoint
60      ;
61 005210 013700 000000G      3$:      MOV      D,R7,R0      ;Get address where break occurred
62 005214 004737 007276'      CALL      LSTADR      ;Display the address
63      ;
64      ;   Display the decoded instruction
65      ;
66 005220 013701 000000G      MOV      D,R7,R1      ;Get address of break
67 005224 004737 007640'      CALL      DECODE      ;Display the decoded instruction
68 005230      TPRINT      #CRLF      ;Go to new line
69      ;
70      ;   Finished
71      ;
72 005240 012601      9$:      MOV      (SP)+,R1
73 005242 000207      RETURN

```

```

1          .SBTTL  FLGBRK -- Set flag if any breakpoints are active
2          ;-----
3          ; FLGBRK is called to determine if there are any active instruction
4          ; breakpoints.  If there are, the D$CKBK flag is set in D.FLAG;
5          ; otherwise the D$CKBK flag is reset.
6          ;
7 005244  010146  FLGBRK: MOV      R1,-(SP)
8          ;
9          ; Determine if there are any normal instruction breakpoints
10         ;
11 005246  012701  000020          MOV      #16.,R1          ;Get index to last breakpoint entry
12 005252  005761  000000G  5$:    TST      D.BKAD(R1)      ;Is this breakpoint set?
13 005256  001007          BNE      6$                      ;Br if yes
14 005260  162701  000002          SUB      #2,R1           ;More to check?
15 005264  002372          BGE      5$                      ;Loop if yes
16         ;
17         ; No breakpoints are active
18         ;
19 005266  042737  000000G 000000G          BIC      #D$CKBK,D.FLAG ;No breakpoints are set
20 005274  000403          BR       9$
21         ;
22         ; There are active breakpoints
23         ;
24 005276  052737  000000G 000000G 6$:    BIS      #D$CKBK,D.FLAG ;Remember some breakpoint is set
25         ;
26         ; Finished
27         ;
28 005304  012601  9$:    MOV      (SP)+,R1
29 005306  000207          RETURN

```

ACRVAL -- Accrue a general value

```

1          .SBTTL  ACRVAL -- Accrue a general value
2          ;-----
3          ; ACRVAL is called to accrue a general value of the form
4          ; [r,]value or $x
5          ; where "r" is a relocation base register number,
6          ; and "x" is an internal register number.
7          ;
8          ; Outputs:
9          ; R0 = Value accrued.
10         ; R1 = Status flag:
11         ;     +1 ==> normal value
12         ;     0 ==> No value gotten.
13         ;     -1 ==> Internal register address.
14         ;
15 005310 010246 ACRVAL: MOV     R2, -(SP)
16         ;
17         ; Get 1st character and see if this is a normal value or a register name.
18         ;
19 005312 004737 005756' CALL    GETCHR      ;Get 1st character
20 005316 120027 000044 CMPB    R0, #'$      ;Is this a register value?
21 005322 001461 BEQ     1$          ;Branch if yes
22 005324 120027 000056 CMPB    R0, #'.      ;"." = Current program counter
23 005330 001451 BEQ     10$         ;Br if period
24 005332 004737 006022' CALL    PSHCHR      ;Save the character
25 005336 120027 000053 CMPB    R0, #'+'      ;Is this a sign character?
26 005342 001414 BEQ     2$          ;Br if yes
27 005344 120027 000055 CMPB    R0, #'-'      ;
28 005350 001411 BEQ     2$          ;
29 005352 120027 000060 CMPB    R0, #'0      ;Is this character a digit?
30 005356 103403 BLO     8$          ;Br if not a digit
31 005360 120027 000071 CMPB    R0, #'9      ;
32 005364 101403 BLOS    2$          ;Br if a digit
33         ;
34         ; No value was specified
35         ;
36 005366 005001 8$:      CLR     R1          ;Return status code
37 005370 005000 CLR     R0          ;Return value of 0
38 005372 000457 BR      9$
39         ;
40         ; Accrue normal value
41         ;
42 005374 004737 005536' 2$:      CALL    ACRNUM      ;Accrue first part of number
43         ;
44         ; See if 1st part of value is a relocation base register number
45         ;
46 005400 010102 MOV     R1, R2      ;Save value that was accrued
47 005402 120027 000054 CMPB    R0, #',      ;Was first part terminated with a comma?
48 005406 001016 BNE     3$          ;Br if not
49 005410 020227 000007 CMP     R2, #7      ;Is first part a valid register number?
50 005414 101404 BLOS    4$          ;Br if ok
51 005416 ERR     #EM$IRB
52 005426 006302 4$:      ASL     R2          ;Convert reg # to word table index
53 005430 004737 005756' CALL    GETCHR      ;Skip over comma
54 005434 004737 005536' CALL    ACRNUM      ;Accrue second part of number
55 005440 066201 000000G ADD     D, RLBS(R2), R1 ;Add relocation base to offset value
56 005444 010100 3$:      MOV     R1, R0      ;Return value in R0
57 005446 012701 000001 MOV     #1, R1      ;Return status code in R1

```

ACRVAL -- Accrue a general value

```

58 005452 000427          BR      9$          ;Return
59
60          ; Value is "." (period). Use current program counter.
61
62 005454 013700 000000G 10$:      MOV      D,R7,R0          ;Get current program counter value (R7)
63 005460 012701 000001      MOV      #1,R1          ;Say we got a normal value
64 005464 000422          BR      9$          ;Return
65
66          ; Internal value ("x")
67
68 005466 004737 005756' 1$:      CALL     GETCHR          ;Get register letter
69 005472 012701 000014      MOV      #NUMINT-1,R1      ;Get # internal registers
70 005476 120061 000364' 6$:      CMPB     RO,INTCHR(R1)      ;Look up register letter
71 005502 001406          BEQ      5$          ;Br if found
72 005504 005301          DEC      R1          ;More to check?
73 005506 002373          BGE      6$          ;Loop if yes
74 005510          ERR      #EM$IIV          ;Invalid register name
75 005520 006301 5$:      ASL      R1          ;Convert index to word table index
76 005522 016100 000402'      MOV      INTADR(R1),R0      ;Get address of cell for register
77 005526 012701 177777      MOV      #-1,R1          ;Get status code
78
79          ; Finished
80
81 005532 012602 9$:      MOV      (SP)+,R2
82 005534 000207          RETURN

```

1			.SBTTL ACRNUM -- Accrue a number
2			-----
3			; ACRNUM is called to accrue a number.
4			; The number may be octal or decimal (with a decimal point).
5			;
6			; Outputs:
7			; R1 = Value accrued
8			; R0 = Delimiter character that was hit.
9			;
10	005536	010246	ACRNUM: MOV R2, -(SP)
11	005540	010346	MOV R3, -(SP)
12	005542	010546	MOV R5, -(SP)
13			;
14			; See if number has a leading sign character
15			;
16	005544	005002	CLR R2 ; Assume result should be positive
17	005546	004737 005756'	CALL GETCHR ; Get 1st character of number
18	005552	120027 000053	CMPB R0, #'+' ; Leading plus sign?
19	005556	001407	BEQ 6\$; Br if yes
20	005560	120027 000055	CMPB R0, #'-' ; Leading minus sign?
21	005564	001002	BNE 8\$; Br if not
22	005566	005202	INC R2 ; Set flag to negate the value
23	005570	000402	BR 6\$
24	005572	004737 006022'	8\$: CALL PSHCHR ; Push the first digit
25			;
26			; Scan the number and store in D.NMBF
27			;
28	005576	012701 000000G	6\$: MOV #D.NMBF, R1 ; Point to number character buffer
29	005602	012705 000010	MOV #8, R5 ; Assume this is an octal value
30	005606	004737 005756'	1\$: CALL GETCHR ; Get next character of number
31	005612	120027 000060	CMPB R0, #'0 ; Is this character a digit?
32	005616	103415	BLO 2\$; Br if not
33	005620	120027 000071	CMPB R0, #'9 ; Decimal digit?
34	005624	101012	BHI 2\$; Br if not
35	005626	120027 000067	CMPB R0, #'7 ; Octal digit?
36	005632	101402	BLOS 3\$; Br if yes
37	005634	012705 000012	MOV #10, R5 ; This must be a decimal value
38	005640	020127 000000G	3\$: CMP R1, #D.NMBE ; Are we about to overflow the buffer?
39	005644	103040	BHIS 11\$; Br if yes
40	005646	110021	MOVB R0, (R1)+ ; Store character into buffer
41	005650	000756	BR 1\$; Loop to get rest of number
42			;
43			; Hit end of number
44			;
45	005652	105011	2\$: CLRB (R1) ; Put null at end of buffer
46	005654	120027 000056	CMPB R0, #'.' ; Decimal point at end of number?
47	005660	001003	BNE 4\$; Br if not
48	005662	012705 000012	MOV #10, R5 ; Remember this is a decimal value
49	005666	000402	BR 5\$
50	005670	004737 006022'	4\$: CALL PSHCHR ; Save delimiter
51			;
52			; Convert number string to binary value
53			;
54	005674	012703 000000G	5\$: MOV #D.NMBF, R3 ; Point to start of buffer
55	005700	005001	CLR R1 ; Store value in R1
56	005702	112300	7\$: MOVB (R3)+, R0 ; Get next character
57	005704	001405	BEQ 9\$; Br if hit end of number

ACRNUM -- Accrue a number

```
58 005706 162700 000060          SUB    #'0,R0          ;Convert digit to binary value
59 005712 070105          MUL     R5,R1          ;Multiply previous value by 8 or 10.
60 005714 060001          ADD     R0,R1          ;Add in new value
61 005716 000771          BR      7$            ;Loop to get rest of number
62                               ;
63                               ; See if we should negate the value
64                               ;
65 005720 005702          9$:    TST     R2          ;Should we negate the value
66 005722 001401          BEQ     10$           ;Br if not
67 005724 005401          NEG     R1          ;Negate the value
68                               ;
69                               ; Finished
70                               ;
71 005726 004737 005756'      10$:  CALL   GETCHR          ;Get delimiter character in R0
72 005732 004737 006022'      CALL   PSHCHR          ;But push it back too
73 005736 012605          MOV     (SP)+,R5
74 005740 012603          MOV     (SP)+,R3
75 005742 012602          MOV     (SP)+,R2
76 005744 000207          RETURN
77                               ;
78                               ; Number overflowed the buffer
79                               ;
80 005746          11$:    ERR      #EM$NTL          ;Number too long
```

GETCHR -- Get next character from terminal

```

1          .SBTTL  GETCHR -- Get next character from terminal
2          ;-----
3          ; Get the next character from the terminal.
4          ;
5          ; Outputs:
6          ;   RO = Character gotten
7          ;
8 005756 GETCHR:
9          ;
10         ; See if PSHCHR was called to save a character
11         ;
12 005756 113700 0000000 MOVB    D.SVCH,RO      ;Has a character been pushed?
13 005762 001014         BNE     9$             ;Br if yes
14         ;
15         ; No saved character.
16         ; Get character from terminal.
17         ;
18 005764         .TTYIN                ;Get character from terminal
19 005770 005237 0000000 INC      D.PCOL        ;Advance print column
20         ;
21         ; Convert lower-case characters to upper-case
22         ;
23 005774 120027 000141     CMPB    RO,#141      ;Lower-case a
24 006000 103405         BLO     9$
25 006002 120027 000172     CMPB    RO,#172      ;Lower-case z
26 006006 101002         BHI     9$
27 006010 162700 000040     SUB     #40,RO       ;Convert lower-case to upper-case
28         ;
29         ; Finished
30         ;
31 006014 105037 0000000 9$: CLRB    D.SVCH      ;No saved character
32 006020 000207         RETURN
33         ;
34         .SBTTL  PSHCHR -- Push a character
35         ;-----
36         ; Push a character so GETCHR will get it on next call.
37         ;
38         ; Inputs:
39         ;   RO = Character to be pushed.
40         ;
41 006022 110037 0000000 PSHCHR: MOVB    RO,D.SVCH    ;Save the character for GETCHR
42 006026 000207         RETURN                ;Finished

```

PRTDCT -- Print octal value

1				.SBTTL PRTDCT -- Print octal value	
2				-----	
3				; Print an octal value with no leading zeroes.	
4					
5				; Inputs:	
6				; R0 = Value to print.	
7					
8	006030	010146		PRTDCT: MOV R1, -(SP)	
9	006032	010246		MOV R2, -(SP)	
10	006034	010346		MOV R3, -(SP)	
11	006036	005003		CLR R3	; Say non-zero digit not seen yet
12	006040	010001		MOV R0, R1	; Get value to print
13	006042	005000		CLR R0	; Get 1st bit into R0
14	006044	073027	000001	ASHC #1, R0	
15	006050	012702	000006	MOV #6, R2	; Print total of 6 digits
16	006054	000403		BR 2\$	
17	006056	005000		1\$: CLR R0	; Get next digit into R0
18	006060	073027	000003	ASHC #3, R0	
19	006064	050003		2\$: BIS R0, R3	; Remember if non-zero digit seen
20	006066	001406		BEQ 3\$	; Br if no non-zero digits seen yet
21	006070	062700	000060	ADD #'0, R0	; Convert to ascii character
22	006074			.TTYOUT	; Print the character
23	006100	005237	0000006	INC D.PCOL	; Advance print column
24	006104	077214		3\$: SOB R2, 1\$	; Loop if more digits to print
25	006106	005703		TST R3	; Did we see any non-zero digits?
26	006110	001006		BNE 4\$	; Br if yes
27	006112			.TTYOUT #'0	; Print one zero if not
28	006122	005237	0000006	INC D.PCOL	; Advance print column
29	006126	012603		4\$: MOV (SP)+, R3	
30	006130	012602		MOV (SP)+, R2	
31	006132	012601		MOV (SP)+, R1	
32	006134	000207		RETURN	

```

1
2
3
4
5
6
7 006136 010146
8 006140 010246
9 006142 010001
10 006144 005000
11 006146 073027 000001
12 006152 012702 000006
13 006156 000403
14 006160 005000
15 006162 073027 000003
16 006166 062700 000060
17 006172
18 006176 005237 000000G
19 006202 077212
20 006204 012602
21 006206 012601
22 006210 000207
23
24
25
26
27
28
29
30
31 006212 010146
32 006214 010246
33 006216 010001
34 006220 000301
35 006222 005000
36 006224 073027 000002
37 006230 012702 000003
38 006234 000403
39 006236 005000
40 006240 073027 000003
41 006244 062700 000060
42 006250
43 006254 005237 000000G
44 006260 077212
45 006262 012602
46 006264 012601
47 006266 000207

```

```

.SBTTL PRTWRD -- Print octal word value
-----
; Print a 16-bit octal value;
; Inputs:
; R0 = Value to be printed.
;
PRTWRD: MOV R1, -(SP)
        MOV R2, -(SP)
        MOV R0, R1 ;Get value to print
        CLR R0 ;Get 1st bit into R0
        ASHC #1, R0
        MOV #6, R2 ;Print total of 6 digits
        BR 2$
1$: CLR R0 ;Get next digit into R0
   ASHC #3, R0
2$: ADD #'0, R0 ;Convert to ascii character
   .TTYOUT ;Print the character
   INC D.PCOL ;Advance print column
   SOB R2, 1$ ;Loop if more digits to print
   MOV (SP)+, R2
   MOV (SP)+, R1
   RETURN

```

```

.SBTTL PRTBYT -- Print octal byte value
-----
; Print an 8-bit octal value.
;
; Inputs:
; R0 = Value to be printed.
;
PRTBYT: MOV R1, -(SP)
        MOV R2, -(SP)
        MOV R0, R1 ;Get value to be printed
        SWAB R1 ;Left-justify value in R1
        CLR R0 ;Move 1st 2 bits into R0
        ASHC #2, R0
        MOV #3, R2 ;Print 3 digits total
        BR 2$
1$: CLR R0 ;Get next octal digit to R0
   ASHC #3, R0
2$: ADD #'0, R0 ;Convert to ascii character
   .TTYOUT ;Print the character
   INC D.PCOL ;Advance print column
   SOB R2, 1$ ;Loop if more digits to print
   MOV (SP)+, R2
   MOV (SP)+, R1
   RETURN

```

```

1
2
3
4
5
6
7
8 006270 010146
9 006272 010246
10
11
12
13 006274 012702 000003
14 006300 005046
15 006302 010001
16 006304 005000
17 006306 071027 000050
18 006312 116146 000220'
19 006316 010001
20 006320 077207
21
22
23
24 006322 012600
25 006324 001405
26 006326
27 006332 005237 000000G
28 006336 000771
29
30
31
32 006340 012602
33 006342 012601
34 006344 000207

```

```

.SBTTL PRTR50 -- Print a RAD50 value
-----
; PRTR50 is called to print a RAD50 (radix 50) value.
;
; Inputs:
; R0 = Value to be printed.
;
PRTR50: MOV     R1, -(SP)
        MOV     R2, -(SP)
;
; Convert value to ascii character string and stack the characters
;
        MOV     #3, R2          ;Get # chars to convert
        CLR     -(SP)          ;Put null on stack to signal end
        MOV     R0, R1          ;Get value to be converted
1$:     CLR     R0              ;Clear high-order for divide
        DIV     #50, R0         ;Divide R0-R1 by 50
        MOVB    R50CHR(R1), -(SP); Convert remainder to ascii and stack it
        MOV     R0, R1          ;Get quotient
        SOB     R2, 1$          ;Loop if more to convert
;
; Finished conversion. Print the result.
;
2$:     MOV     (SP)+, R0        ;Get char to print
        BEQ     3$              ;Br if hit end of string
        .TTYOUT                  ;Print the character
        INC     D.PCOL          ;Advance print column
        BR      2$              ;Loop to print more
;
; Finished
;
3$:     MOV     (SP)+, R2
        MOV     (SP)+, R1
        RETURN

```

ERRPRT -- Print an error message

```

1          .SBTTL  ERRPRT -- Print an error message
2          ;-----
3          ; Print the error message whose address is in R1 and then go and
4          ; get a new command.
5          ;
6 006346  ERRPRT: TPRINT  #CRLF          ;Go to new line
7 006356          TPRINT  R1            ;Print error message
8 006364 005037 000000G          CLR      D.PCOL          ;Say print column = 0
9 006370 013706 000000G          MOV      D.SPSV,SP        ;Reset stack pointer
10 006374 000137 001510'         JMP      NEWCMD           ;Go get a new command
11
12          .SBTTL  LSTTXT -- Print text string
13          ;-----
14          ; LSTTXT is called by use of the PRINT macro.  It prints a text
15          ; string on the terminal without carriage-return, line-feed.
16          ;
17          ; Inputs:
18          ; R2 = Address of string in ASCIZ form that follows call.
19          ;
20 006400 112200  LSTTXT: MOVB      (R2)+,R0          ;Get next char from text string
21 006402 001405          BEQ      2$                ;Br if hit end of string
22 006404          .TTYOUT          ;Print the character
23 006410 005237 000000G          INC      D.PCOL          ;Advance print column
24 006414 000771          BR       LSTTXT             ;Loop till end of string hit
25 006416 005202          2$: INC      R2              ;Bound up to even byte address
26 006420 042702 000001          BIC      #1,R2
27 006424 000202          RTS      R2                ;Return following string

```

SETLOC -- Set address of open cell

```

1          .SBTTL  SETLOC -- Set address of open cell
2          ;-----
3          ; SETLOC is called to set the address of the currently open cell.
4          ;
5          ; Inputs:
6          ;   D. VAL1 = Address of cell to open.
7          ;   D. V1FL = Mode of address (+1=Cell in user's job, -1=Internal cell)
8          ;
9          ; Outputs:
10         ;   D. LOC  = Address of open cell.
11         ;   D. LOCM = Mode of cell (+1 / -1)
12         ;
13 006426 113700 000000G SETLOC: MOVB   D. V1FL,RO      ;Get mode of address
14 006432 001405         BEQ     9$                    ;Br if no address specified
15 006434 110037 000000G         MOVB   RO,D. LOCM      ;Set mode of current cell
16 006440 013737 000000G 000000G        MOV    D. VAL1,D. LOC    ;Set address of current cell
17 006446 000207         9$:   RETURN

```

```

1          .SBTTL  STRVAL -- Store value into open cell
2          ;-----
3          ; STRVAL is called to store a value into the currently open cell.
4          ;
5          ; Inputs:
6          ;   RO      = Value to store.
7          ;   D.LOC    = Address where to store value.
8          ;   D.LOCM   = +1=Store into user's area, -1=Store into internal register.
9          ;   D.BYTM   = 0=Word mode store, 1=Byte mode store.
10         ;
11 006450    010146    STRVAL: MOV      R1,-(SP)
12 006452    010246          MOV      R2,-(SP)
13         ;
14         ; Make sure a cell is open
15         ;
16 006454    105737    000000G      TSTB     D.LOCM          ;Is a cell open now?
17 006460    001512          BEQ      9$                  ;Br if not
18         ;
19         ; Determine if we are storing into user's job or internal cell
20         ;
21 006462    105737    000000G      TSTB     D.LOCM          ;Internal or user job?
22 006466    100457          BMI      1$                  ;Br if internal cell
23         ;
24         ; Store into user's job
25         ;
26 006470    105737    000000G      TSTB     D.BYTM          ;Byte or word mode?
27 006474    001013          BNE      2$                  ;Br if byte mode
28         ;
29         ; Store into word
30         ;
31 006476    010046          MOV      RO,-(SP)              ;Put value on stack
32 006500    013700          MOV      D.LOC,R0              ;Get address where we should store
33 006504    032737    000000G 000000G      BIT      #D$ISPC,D.FLAG ;Should we use I-space?
34 006512    001402          BEQ      4$                  ;Br if not (D-space is default)
35 006514    006610          MTPI     (R0)                  ;Store into program I-space
36 006516    000473          BR       9$
37 006520    106610    4$: MTPD      (R0)                  ;Store into user's area
38 006522    000471          BR       9$
39         ;
40         ; Store into byte
41         ;
42 006524    042700    177400    2$: BIC      #^C377,R0      ;Clear high-order byte of value being stored
43 006530    013702    000000G      MOV      D.LOC,R2      ;Get address where we are to store
44 006534    042702    000001      BIC      #1,R2          ;Force address to be even
45 006540    032737    000000G 000000G      BIT      #D$ISPC,D.FLAG ;I-space reference?
46 006546    001402          BEQ      7$                  ;Br if not
47 006550    006512          MFPI     (R2)                  ;Get current I-space value
48 006552    000401          BR       8$                  ;And continue
49 006554    106512    7$: MFPD      (R2)                  ;Get current contents of word (D-space)
50 006556    032737    000001 000000G 8$: BIT      #1,D.LOC ;Even or odd byte?
51 006564    001004          BNE      3$                  ;Br if odd byte
52 006566    042716    000377      BIC      #377,(SP)      ;Clear low-order byte of old value
53 006572    050016          BIS      R0,(SP)              ;Put in new low-order byte
54 006574    000404          BR       5$                  ;Go set into user space
55 006576    042716    177400    3$: BIC      #177400,(SP) ;Clear high-order byte
56 006602    000300          SWAB     R0                  ;Get new value to high-order byte
57 006604    050016          BIS      R0,(SP)              ;Store new high-order byte

```

STRVAL -- Store value into open cell

```

58 006606 032737 000000G 000000G 5$:    BIT    #D$ISPC,D.FLAG ; Store into I-space?
59 006614 001402                      BEQ     6$          ; Br if not (D-space is default)
60 006616 006612                      MTPI   (R2)         ; Store into user I-space
61 006620 000432                      BR      9$
62 006622 106612          6$:    MTPD    (R2)         ; Store into user D-space
63 006624 000430                      BR      9$
64                                     ;
65                                     ; Store into internal register
66                                     ;
67 006626 105737 000000G          1$:    TSTB    D.BYTM         ; Byte or word mode?
68 006632 001003                      BNE     12$          ; Br if byte mode
69                                     ;
70                                     ; Store into word
71                                     ;
72 006634 010077 000000G                      MOV     RO,@D.LOC ; Store into internal cell
73 006640 000422                      BR      9$
74                                     ;
75                                     ; Store into byte
76                                     ;
77 006642 042700 177400          12$:    BIC     #^C377,RO      ; Clear high-order byte of value being stored
78 006646 013702 000000G                      MOV     D.LOC,R2   ; Get address where we are to store
79 006652 042702 000001                      BIC     #1,R2      ; Force address to be even
80 006656 032737 000001 000000G          BIT     #1,D.LOC      ; Even or odd byte?
81 006664 001344                      BNE     3$          ; Br if odd byte
82 006666 042712 000377                      BIC     #377,(R2)   ; Clear low-order byte of old value
83 006672 050012                      BIS     RO,(R2)         ; Put in new low-order byte
84 006674 000404                      BR      9$
85 006676 042712 177400          13$:    BIC     #177400,(R2)  ; Clear high-order byte
86 006702 000300                      SWAB    RO              ; Get new value to high-order byte
87 006704 050012                      BIS     RO,(R2)         ; Store new high-order byte
88                                     ;
89                                     ; Finished
90                                     ;
91 006706 012602          9$:    MOV     (SP)+,R2
92 006710 012601                      MOV     (SP)+,R1
93 006712 000207                      RETURN

```

GETVAL -- Get value from current cell

```

1          .SBTTL  GETVAL -- Get value from current cell
2          ;-----
3          ; GETVAL is called to get the value from the currently open cell.
4          ;
5          ; Inputs:
6          ;   D.LOC  = Address of cell from which value is to be fetched.
7          ;   D.LOCM = +1=fetch from user's area, -1=Fetch from internal register.
8          ;   D.BYTM = 0=Word mode, 1=Byte mode
9          ;
10         ; Outputs:
11         ;   RO = Value obtained
12         ;
13 006714 010246 GETVAL: MOV      R2,-(SP)
14         ;
15         ; Determine if we are fetching from user's job or internal cell
16         ;
17 006716 105737 000000G      TSTB    D.LOCM      ;Internal or user job?
18 006722 100442      BMI     1$              ;Br if internal cell
19         ;
20         ; Fetch from user's job
21         ;
22 006724 105737 000000G      TSTB    D.BYTM      ;Byte or word mode?
23 006730 001013      BNE     2$              ;Br if byte mode
24         ;
25         ; Fetch from word
26         ;
27 006732 013700 000000G      MOV     D.LOC,R0    ;Get address of value wanted
28 006736 032737 000000G 000000G      BIT     #D$ISPC,D.FLAG ;Should we use I-space?
29 006744 001402      BEQ     4$              ;Br if not (D-space default)
30 006746 006510      MFPI    (R0)            ;Get value from user I-space
31 006750 000401      BR      5$              ;And continue
32 006752 106510 4$: MFPD    (R0)            ;Get value from user D-space
33 006754 012600 5$: MOV     (SP)+,R0        ;Return in RO
34 006756 000446      BR      9$
35         ;
36         ; Fetch from byte
37         ;
38 006760 013702 000000G 2$: MOV     D.LOC,R2    ;Get address of word
39 006764 042702 000001      BIC     #1,R2      ;Force address to be even
40 006770 032737 000000G 000000G      BIT     #D$ISPC,D.FLAG ;Should we use I-space
41 006776 001402      BEQ     6$              ;Br if not (D-space default)
42 007000 006512      MFPI    (R2)            ;Get current value from I-space
43 007002 000401      BR      7$              ;And continue
44 007004 106512 6$: MFPD    (R2)            ;Get current value from D-space
45 007006 012600 7$: MOV     (SP)+,R0        ;Get word value to RO
46 007010 032737 000001 000000G      BIT     #1,D.LOC      ;Even or odd byte?
47 007016 001401      BEQ     3$              ;Br if low-order byte wanted
48 007020 000300      SWAB    R0              ;Swap bytes if high-order byte wanted
49 007022 042700 177400 3$: BIC     #177400,R0 ;Clear high-order byte
50 007026 000422      BR      9$
51         ;
52         ; Fetch from internal register
53         ;
54 007030 105737 000000G 1$: TSTB    D.BYTM      ;Byte or word mode?
55 007034 001003      BNE     12$             ;Br if byte mode
56         ;
57         ; Fetch from word

```

GETVAL -- Get value from current cell

```
58
59 007036 017700 000000G      ;      MOV      @D.LOC,R0      ;Get the value
60 007042 000414              ;      BR       9$
61                          ;
62                          ;  Fetch from byte
63                          ;
64 007044 013702 000000G      12$:  MOV      D.LOC,R2      ;Get address where we are to fetch
65 007050 042702 000001              BIC      #1,R2        ;Force address to be even
66 007054 011200              MOV      (R2),R0      ;Get the word value
67 007056 032737 000001 000000G  BIT      #1,D.LOC      ;Even or odd byte?
68 007064 001401              BEQ      13$          ;Br if even byte wanted
69 007066 000300              SWAB     R0           ;Swap high-order byte to low-order
70 007070 042700 177400      13$:  BIC      #177400,R0    ;Clear high-order byte
71                          ;
72                          ;  Finished
73                          ;
74 007074 012602      9$:  MOV      (SP)+,R2
75 007076 000207              RETURN
```

SHOLOC -- Display current address

```

1          .SBTTL  SHOLOC -- Display current address
2          ;-----
3          ; Display address of currently open cell.
4          ; The address may be either in the user's program space
5          ; or may be the address of an internal register ($x).
6          ;
7          ; Inputs:
8          ;   D.LOC  = Address to be displayed
9          ;   D.LOCM = Mode of address (+1==>In user's program, -1==>Internal cell)
10         ;
11 007100    010146
12 007102    010246
13 007104    010346
14 007106    010446
15
16         ; Determine if this is an internal or external cell
17         ;
18 007110    105737    000000G
19 007114    002405
20
21         ; Display address of cell in user's program
22         ;
23 007116    013700    000000G
24 007122    004737    007276'
25 007126    000456
26
27         ; Display name of internal cell
28         ;
29 007130    013700    000000G
30 007134    012701    000402'
31 007140    012702    177777
32 007144    005003
33 007146    020011
34 007150    103406
35 007152    010004
36 007154    161104
37 007156    020402
38 007160    101002
39 007162    010402
40 007164    010103
41 007166    062701    000002
42 007172    020127    000434'
43 007176    103763
44
45         ; Display register name
46         ;
47 007200    162703    000402'
48 007204    006203
49 007206
50 007216    005237    000000G
51 007222    116300    000364'
52 007226
53 007232    005237    000000G
54
55         ; Display offset from register
56         ;
57 007236    005702

```

```

          SHOLOC: MOV     R1,-(SP)
                  MOV     R2,-(SP)
                  MOV     R3,-(SP)
                  MOV     R4,-(SP)
;
; Determine if this is an internal or external cell
;
          TSTB     D.LOCM          ; Internal or external address?
          BLT      1$              ; Br if internal address
;
; Display address of cell in user's program
;
          MOV      D.LOC,R0        ; Get address to be displayed
          CALL     LSTADR          ; Display relative to relocation base
          BR       9$
;
; Display name of internal cell
;
1$:        MOV     D.LOC,R0        ; Get address of internal cell
          MOV     #INTADR,R1      ; Point to table of addresses of cells
          MOV     #-1,R2          ; Init for search
          CLR     R3
2$:        CMP     R0,(R1)         ; Is address below base of this cell?
          BLO     3$              ; Br if yes
          MOV     R0,R4           ; Get address of interest
          SUB     (R1),R4         ; Determine distance from cell base addr
          CMP     R4,R2           ; Is this the closest we've seen so far?
          BHI     3$              ; Br if not
          MOV     R4,R2           ; Remember offset from nearest cell
          MOV     R1,R3           ; Remember address of nearest cell
3$:        ADD     #2,R1           ; Point to address of next cell
          CMP     R1,#INTEND      ; Checked all cells?
          BLO     2$              ; Loop if not
;
; Display register name
;
          SUB     #INTADR,R3      ; Get offset into address table
          ASR     R3              ; Convert to byte table index
          .TTYOUT #'$            ; Print "$"
          INC     D.PCOL          ; Advance print column
          MOVB    INTCHR(R3),R0   ; Get name of internal register
          .TTYOUT          ; Print register name
          INC     D.PCOL          ; Advance print column
;
; Display offset from register
;
          TST     R2              ; Any offset from base of cell?

```

SHOLOC -- Display current address

58	007240	001411		BEQ	9\$		;Br if not
59	007242			. TTYOUT	#'+		;Print "+"
60	007252	005237	000000G	INC	D.PCOL		;Advance print column
61	007256	010200		MOV	R2,R0		;Get offset value
62	007260	004737	006030'	CALL	PRT OCT		;Print octal value for offset
63							
64							
65							
66	007264	012604		9\$: MOV	(SP)+,R4		
67	007266	012603		MOV	(SP)+,R3		
68	007270	012602		MOV	(SP)+,R2		
69	007272	012601		MOV	(SP)+,R1		
70	007274	000207		RETURN			

; Finished

LSTADR -- Display address in user's program

```

1          .SBTTL  LSTADR -- Display address in user's program
2          ;-----
3          ; LSTADR is called to display an address in the user's program.
4          ; LSTADR functions the same as SHOADR except that LSTADR checks the
5          ; DP$LAA flag in the $F format register. If the DP$LAA flag is cleared
6          ; the address is displayed in relative form; if it is set, it is displayed
7          ; in absolute form.
8          ;
9          ; Inputs:
10         ; RO = Address to be displayed.
11         ;
12         LSTADR:
13         ;
14         ; See if DP$LAA is set in $F register
15         ;
16         007276 032737 000000G 000000G      BIT      #DP$LAA,D. PFMT  ;Absolute addresses wanted?
17         007304 001003                      BNE      1$              ;Br if yes
18         ;
19         ; Display address with relocation base
20         ;
21         007306 004737 007322'              CALL      SHOADR          ;Display address with relocation base
22         007312 000402                      BR        9$
23         ;
24         ; Display absolute address
25         ;
26         007314 004737 006030'              1$:      CALL      PRTOCT      ;Display absolute address
27         ;
28         ; Finished
29         ;
30         007320 000207                      9$:      RETURN

```

SHOADR -- Display address in user's program

```

1          .SBTTL  SHOADR -- Display address in user's program
2          ;-----
3          ; SHOADR is called to display an address in the user's program.
4          ; A check is made to determine which relocation register the address
5          ; is relative to and if any, the address is printed as "r,o".
6          ;
7          ; Inputs:
8          ;   RO = Address to display.
9          ;
10         SHOADR: MOV     R1, -(SP)
11                MOV     R2, -(SP)
12                MOV     R3, -(SP)
13                MOV     R4, -(SP)
14          ;
15          ; Determine if address is relative to a relocation register
16          ;
17         MOV     RO, R4          ; Save original address
18         CALL    RELADR          ; Determine if addr is relative to reloc regn
19         BCS     4$              ; Br if not
20          ;
21          ; Print relocation register number
22          ;
23         SUB     D, RLBS(R3), R4 ; Compute offset within the region
24         MOV     R3, RO          ; Get register index
25         ASR     RO              ; Convert index to number
26         CALL    PRTOCT          ; Print register number
27         .TTYOUT #54             ; Print ", "
28         INC     D, PCOL         ; Advance print column
29          ;
30          ; Print offset within region
31          ;
32         4$: MOV     R4, RO          ; Get offset within region
33         CALL    PRTOCT          ; Print the value
34          ;
35          ; Finished
36          ;
37         MOV     (SP)+, R4
38         MOV     (SP)+, R3
39         MOV     (SP)+, R2
40         MOV     (SP)+, R1
41         RETURN

```

RELADR -- Determine if address is in relocation region

```

1          .SBTTL  RELADR -- Determine if address is in relocation region
2          ;-----
3          ; RELADR is called to determine if an address is within a relocation
4          ; region.
5          ;
6          ; Inputs:
7          ;   R0 = Address to be checked.
8          ;
9          ; Outputs:
10         ;   C-flag cleared ==> Address is within a relocation region.
11         ;   C-flag set    ==> Address is not within a relocation region.
12         ;   R3 = Index to relocation region to be used with address.
13         ;
14 007412 010146
15 007414 010246
16 007416 010446
17
18         ; Search for relocation region defined below the address
19         ;
20 007420 005001          CLR      R1          ; Init relocation register index
21 007422 012702 177777  MOV      #-1,R2      ; Init closest offset value
22 007426 010203          MOV      R2,R3      ; Say no relocation base found yet
23 007430 005761 000000G  2$:    TST      D,RLBS(R1) ; Is this relocation register in use?
24 007434 001412          BEQ      3$         ; Br if not
25 007436 020061 000000G  CMP      R0,D,RLBS(R1) ; Is address above this relocation base?
26 007442 103407          BLO      3$         ; Br if not
27 007444 010004          MOV      R0,R4      ; Get address of interest
28 007446 166104 000000G  SUB      D,RLBS(R1),R4 ; Calculate offset within region
29 007452 020402          CMP      R4,R2      ; Is this the closest one so far?
30 007454 101002          BHI      3$         ; Br if not
31 007456 010402          MOV      R4,R2      ; Remember smallest offset
32 007460 010103          MOV      R1,R3      ; Remember relocation register index
33 007462 062701 000002  3$:    ADD      #2,R1  ; Advance relocation register index
34 007466 020127 000016  CMP      R1,#14.    ; Checked all relocation registers?
35 007472 101756          BLOS     2$         ; Br if not
36
37         ; See if we found a relocation region
38         ;
39 007474 005703          TST      R3          ; Did we find a relocation region
40 007476 002002          BGE      5$         ; Br if yes
41 007500 000261          SEC          ; Say no region found
42 007502 000401          BR       9$         ;
43 007504 000241  5$:    CLC          ; Say a region was found
44
45         ; Finished
46         ;
47 007506 012604  9$:    MOV      (SP)+,R4
48 007510 012602          MOV      (SP)+,R2
49 007512 012601          MOV      (SP)+,R1
50 007514 000207          RETURN

```

DOPRT -- Print a text string

```

1          .SBTTL  DOPRT  -- Print a text string
2          ;-----
3          ; DOPRT is called to print a text string.
4          ; It directly calls routines in TSTTY to print each character.
5          ;
6          ; Inputs:
7          ;   RO = Pointer to asciz string.
8          ;
9 007516    010146
10 007520    010246
11          ;
12          ; Set up pointer to start of string
13          ;
14 007522    010002
15 007524    113701    000000G
16          ;
17          ; Get each char out of the string and print it
18          ;
19 007530    112200
20 007532    001411
21 007534    120027    000200
22 007540    001422
23 007542
24 007550    005237    000000G
25 007554    000765
26          ;
27          ; Print Carriage-return Line-feed at end of string
28          ;
29 007556    012700    000015
30 007562
31 007570    012700    000012
32 007574
33 007602    005037    000000G
34          ;
35          ; Finished
36          ;
37 007606    012602
38 007610    012601
39 007612    000207

          DOPRT:  MOV      R1, -(SP)
                  MOV      R2, -(SP)
                  ;
                  ; Set up pointer to start of string
                  ;
                  MOV      RO, R2          ;Get pointer to start of string
                  MOVB     CORUSR, R1      ;Get current job index number
                  ;
                  ; Get each char out of the string and print it
                  ;
1$:             MOVB     (R2)+, RO          ;Get next char from string
                  BEQ      2$              ;Br if hit null at end
                  CMPB     RO, #200        ;Is string terminated with a 200?
                  BEQ      9$              ;Br if yes
                  OCALL    PUTCHR          ;Print the character
                  INC      D.PCOL          ;Advance print column
                  BR        1$              ;Loop to print rest of string
                  ;
                  ; Print Carriage-return Line-feed at end of string
                  ;
2$:             MOV      #CR, RO          ;Get carriage return
                  OCALL    PUTCHR          ;Print it
                  MOV      #LF, RO        ;Get line feed
                  OCALL    PUTCHR          ;Print it
                  CLR      D.PCOL          ;Say we are back to column 1
                  ;
                  ; Finished
                  ;
9$:             MOV      (SP)+, R2
                  MOV      (SP)+, R1
                  RETURN

```

```
1          .SBTTL  DBGON  -- Place terminal in debug mode
2          ;-----
3          ;  DBGON is called to place the terminal in debug mode.
4          ;
5 007614    012700    000002'  DBGON:  MOV      #ONEMT,RO
6 007620    104375              EMT      375
7 007622    000207              RETURN
8
9          .SBTTL  DBGOFF -- Turn off terminal debug mode
10         ;-----
11         ;  DBGOFF is called to take the terminal out of debug mode
12         ;
13 007624    012700    000004'  DBGOFF: MOV      #OFFEMT,RO
14 007630    104375              EMT      375
15 007632    000207              RETURN
16
17         .SBTTL  CHKADR -- Determine if address is valid
18         ;-----
19         ;  CHKADR is called to determine if an address is within the
20         ;  range of the user's program space.
21         ;
22         ;  Inputs:
23         ;  RO = Address to be checked.
24         ;
25         ;  Outputs:
26         ;  C-flag cleared ==> address is ok
27         ;  C-flag set    ==> address is invalid
28         ;
29 007634    000241  CHKADR:  CLC
30 007636    000207              RETURN
```

```

1          .SBTTL  DECODE -- Decode instruction into symbolic form
2          ;-----
3          ; DECODE is called to decode an instruction word into its symbolic form.
4          ; The symbolic form of the instruction is displayed on the console.
5          ;
6          ; Inputs:
7          ;   R1 = Address of instruction to be decoded.
8          ;
9          ; Outputs:
10         ;   D. ILEN = Length (in bytes) of decoded instruction and operands.
11         ;
12 007640 010146 DECODE: MOV      R1,-(SP)
13 007642 010246      MOV      R2,-(SP)
14 007644 010346      MOV      R3,-(SP)
15 007646 010446      MOV      R4,-(SP)
16 007650 010546      MOV      R5,-(SP)
17         ;
18         ; Initialize instruction length to 2 bytes
19         ;
20 007652 012737 000002 000000G      MOV      #2,D. ILEN      ; Say base instruction length = 2 bytes
21         ;
22         ; Initialize column counter for decoded instruction display
23         ;
24 007660      PRINT      <      >      ; First print a few spaces
25 007672 005037 000000G      CLR      D. PCOL      ; Say we are at column 1 now
26         ;
27         ; Fetch the instruction word
28         ;
29 007676 006521      MFPI      (R1)+      ; Get the instruction from user's space
30 007700 012602      MOV      (SP)+,R2      ; Get instruction off of stack
31         ;
32         ; Search the instruction list for this instruction
33         ;
34 007702 012704 012114'      MOV      #INSBAS,R4      ; Point to base of instruction decode table
35 007706 010203 1$:      MOV      R2,R3      ; Get instruction to be decoded
36 007710 046403 000000      BIC      IB$MSK(R4),R3      ; Clear operand fields in instruction
37 007714 020364 000002      CMP      R3,IB$VAL(R4)      ; Is this the instruction?
38 007720 001411      BEQ      5$      ; Br if found the instruction
39 007722 116400 000005      MOVB     IB$LEN(R4),R0      ; Get length of ascii name string
40 007726 005200      INC      R0      ; Bound up to word boundary
41 007730 042700 000001      BIC      #1,R0
42 007734 062700 000006      ADD      #IB$NAM,R0      ; Add size of base portion of block
43 007740 060004      ADD      R0,R4      ; Point to next instruction decode block
44 007742 000761      BR      1$      ; Continue searching for the instruction
45         ;
46         ; We have found the instruction in the instruction decode table.
47         ; R4 = Address of instruction decode block.
48         ; Display the symbolic name of the instruction.
49         ;
50 007744 116403 000005 5$:      MOVB     IB$LEN(R4),R3      ; Get length of instruction mnemonic
51 007750 060337 000000G      ADD      R3,D. PCOL      ; Advance column counter
52 007754 010405      MOV      R4,R5      ; Point to the mnemonic ascii string
53 007756 062705 000006      ADD      #IB$NAM,R5
54 007762 112500 6$:      MOVB     (R5)+,R0      ; Get next char of mnemonic
55 007764      .TTYOUT      ; Print the character
56 007770 077304      SOB      R3,6$      ; Loop if more chars to print
57         ;

```

DECODE -- Decode instruction into symbolic form

```
58      ; Now mask out the instruction code from the instruction word leaving
59      ; only the operand fields.
60      ;
61 007772 016400 000000      MOV      IB$MSK(R4),R0      ;Get mask value
62 007776 005100      COM      R0      ;Complement it to leave the operand fields
63 010000 040002      BIC      R0,R2      ;Mask out all but the operand fields
64      ;
65      ; Now call appropriate routine to display the operand values for
66      ; this instruction.
67      ; R1 = Address of word past instruction.
68      ; R2 = Operand field values (instruction word with instruction code masked)
69      ;
70 010002 116400 000004      MOVB     IB$TYP(R4),R0      ;Get the operand type code for this inst
71 010006 004770 014330'    CALL     @TYPVEC(R0)      ;Call appropriate operand display routine
72      ;
73      ; Finished
74      ;
75 010012 012605      MOV      (SP)+,R5
76 010014 012604      MOV      (SP)+,R4
77 010016 012603      MOV      (SP)+,R3
78 010020 012602      MOV      (SP)+,R2
79 010022 012601      MOV      (SP)+,R1
80 010024 000207      RETURN
```

ODCxxx -- Display instruction operands

```

1          .SBTTL  ODCxxx -- Display instruction operands
2          ;-----
3          ;  TYP1 -- Instruction has no operands.
4          ;
5 010026  000207  ODC1:  RETURN
6          ;-----
7          ;
8          ;  TYP2 -- Operand has a single general operand.
9          ;
10 010030  ODC2:
11          ;
12          ;  Tab over to operand field
13          ;
14 010030  012700  000010  MOV      #COLOPN,R0      ;Get operand column number
15 010034  004737  012064'  CALL      ODCTAB        ;Tab over to operand field
16          ;
17          ;  Display the operand
18          ;
19 010040  010200  MOV      R2,R0      ;Get the operand code
20 010042  004737  011000'  CALL      ODCGEN        ;Display general operand
21          ;
22          ;  Finished
23          ;
24 010046  000207  RETURN
25          ;-----
26          ;
27          ;  TYP3 -- Single register operand.
28          ;
29 010050  ODC3:
30          ;
31          ;  Tab over to the operand field
32          ;
33 010050  012700  000010  MOV      #COLOPN,R0      ;Get operand column number
34 010054  004737  012064'  CALL      ODCTAB        ;Tab over to operand field
35          ;
36          ;  Display the register
37          ;
38 010060  010200  MOV      R2,R0      ;Get the register number
39 010062  004737  011472'  CALL      ODCREG        ;Display the register number
40 010066  000207  RETURN

```

ODCxxx -- Display instruction operands

```

1      ; -----
2      ;  TYP4 -- 4-bit absolute numeric operand
3      ;
4 010070 ODC4:
5      ;
6      ;  Tab over to the operand field
7      ;
8 010070 012700 000010      MOV      #COLPN,R0      ;Get operand column number
9 010074 004737 012064'    CALL      ODCTAB      ;Tab over to operand field
10     ;
11     ;  Display the value
12     ;
13 010100 010200      MOV      R2,R0      ;Get value to display
14 010102 004737 006030'  CALL      PRTDCT      ;Print the value
15     ;
16     ;  Finished
17     ;
18 010106 000207      RETURN
19     ; -----
20     ;
21     ;  TYP5 -- Condition code modification instruction
22     ;
23 010110 ODC5:
24     ;
25     ;  Determine which condition code is affected
26     ;
27 010110 032702 000001      BIT      #1,R2      ;C-flag?
28 010114 001404      BEQ      1$      ;Br if not
29 010116      .TTYOUT #'C
30 010126 032702 000002 1$: BIT      #2,R2      ;V-flag?
31 010132 001404      BEQ      2$      ;Br if not
32 010134      .TTYOUT #'V
33 010144 032702 000004 2$: BIT      #4,R2      ;Z-flag?
34 010150 001404      BEQ      3$      ;Br if not
35 010152      .TTYOUT #'Z
36 010162 032702 000010 3$: BIT      #10,R2     ;N-flag?
37 010166 001404      BEQ      4$      ;Br if not
38 010170      .TTYOUT #'N
39     ;
40     ;  Finished
41     ;
42 010200 005237 000000G 4$: INC      D.PCOL      ;Advance column number
43 010204 000207      RETURN

```

ODCxxx -- Display instruction operands

```

1
2 ; -----
3 ; TYP6 -- Branch offset operand
4 ;
5 ODC6:
6 ;
7 ; Tab over to the operand field
8 ;
9      MOV      #COLOPN,R0      ;Get operand column number
10     CALL     ODCTAB          ;Tab over to operand field
11 ;
12 ; Display the branch target
13 ;
14     MOV      R2,R0           ;Get instruction word
15     BIC      #^C377,R0       ;Leave only branch offset
16     BIT      #200,R0         ;Is offset negative?
17     BEQ      1$             ;Br if not
18     BIS      #177400,R0       ;Sign extend the offset
19 1$: ASL      R0               ;Convert word offset to byte offset
20     ADD      R1,R0           ;Add current PC address
21     CALL     ODCADR          ;Display the address
22 ;
23 ; Finished
24 ;
25     RETURN
26 ; -----
27 ; TYP7 -- Register and general destination.
28 ;
29 ODC7:
30 ;
31 ; Tab over to the operand field
32 ;
33     MOV      #COLOPN,R0       ;Get operand column number
34     CALL     ODCTAB          ;Tab over to operand field
35 ;
36 ; Display the register
37 ;
38     MOV      R2,R0           ;Get instruction word
39     ASH      #-6,R0           ;Right justify the register value
40     CALL     ODCREG          ;Display the register name
41     .TTYOUT  #COMMA          ;Display ","
42     INC      D.PCOL          ;Inc column counter
43 ;
44 ; Display the destination address
45 ;
46     MOV      R2,R0           ;Get the destination address descriptor
47     CALL     ODCGEN          ;Display the address
48 ;
49 ; Finished
50 ;
51     RETURN

```

ODCxxx -- Display instruction operands

```

1      ; -----
2      ;  TYP8 -- Register destination and general source operand (SS,R)
3      ;
4 010316 ODC8:
5      ;
6      ;  Tab over to the operand field
7      ;
8 010316 012700 000010      MOV      #COLOPN,R0      ;Get operand column number
9 010322 004737 012064'    CALL      ODC8TAB      ;Tab over to operand field
10     ;
11     ;  Display the source item
12     ;
13 010326 010200      MOV      R2,R0      ;Get the source operand code
14 010330 004737 011000'    CALL      ODCGEN      ;Display it
15 010334      .TTYOUT #COMMA      ;Display ","
16 010344 005237 000000G    INC      D.PCOL      ;Advance column counter
17     ;
18     ;  Display the register destination
19     ;
20 010350 010200      MOV      R2,R0      ;Get the register number
21 010352 072027 177772    ASH      #-6,R0      ;Right justify
22 010356 004737 011472'    CALL      ODCREG      ;Display the register
23     ;
24     ;  Finished
25     ;
26 010362 000207      RETURN

```

ODCxxx -- Display instruction operands

```

1      ; -----
2      ;  TYP9 -- Two general operands
3      ;
4 010364 ODC9:
5      ;
6      ;  Tab over to the operand field
7      ;
8 010364 012700 000010      MOV    #COLOPN,R0      ;Get operand column number
9 010370 004737 012064'    CALL    ODCTAB      ;Tab over to operand field
10     ;
11     ;  Display the source operand
12     ;
13 010374 010200      MOV    R2,R0      ;Get instruction word
14 010376 072027 177772    ASH    #-6,R0      ;Right justify the source item
15 010402 004737 011000'    CALL    ODCGEN      ;Display source item
16 010406      .TTYOUT #COMMA      ;Print a comma
17     ;
18     ;  Display the destination operand
19     ;
20 010416 010200      MOV    R2,R0      ;Get destination item code
21 010420 004737 011000'    CALL    ODCGEN      ;Display destination item
22     ;
23     ;  Finished
24     ;
25 010424 000207      RETURN

```

ODCxxx -- Display instruction operands

```

1      ; -----
2      ;   TYP10 -- SOB instruction
3      ;
4 010426 ODC10:
5      ;
6      ;   Tab over to the operand field
7      ;
8 010426 012700 000010      MOV      #COLOPN,R0      ;Get operand column number
9 010432 004737 012064'    CALL      ODC TAB      ;Tab over to operand field
10     ;
11     ;   Display the register number
12     ;
13 010436 010200      MOV      R2,R0      ;Get instruction word
14 010440 072027 177772    ASH      #-6,R0      ;Right justify register number
15 010444 004737 011472'    CALL      ODC REG      ;Display the register name
16 010450      .TTYOUT #COMMA      ;Print comma
17 010460 005237 000000G    INC      D.PCOL      ;Advance column number
18     ;
19     ;   Display the destination address
20     ;
21 010464 042702 177700      BIC      #^C77,R2      ;Mask out all but branch offset
22 010470 010100      MOV      R1,R0      ;Get the current PC address
23 010472 006302      ASL      R2      ;Get 2* offset
24 010474 160200      SUB      R2,R0      ;Get destination address
25 010476 004737 011720'    CALL      ODCADR      ;Display the address
26     ;
27     ;   Finished
28     ;
29 010502 000207      RETURN

```

ODCxxx -- Display instruction operands

```

1      ; -----
2      ; TYP11 -- EMT and TRAP instructions
3      ;
4 010504 ODC11:
5      ;
6      ; Tab over to the operand field
7      ;
8 010504 012700 000010      MOV    #COLOPN,R0      ;Get operand column number
9 010510 004737 012064'    CALL    ODCTAB      ;Tab over to operand field
10     ;
11     ; Display the argument value
12     ;
13 010514 010200      MOV    R2,R0      ;Get instruction word
14 010516 042700 177400    BIC    #^C377,R0      ;Mask out all but operand value
15 010522 004737 006030'    CALL    PRTDCT      ;Print octal value
16     ;
17     ; Finished
18     ;
19 010526 000207      RETURN
20     ; -----
21     ;
22     ; TYP12 -- Single floating-point operand
23     ;
24 010530 ODC12:
25     ;
26     ; Tab over to the operand field
27     ;
28 010530 012700 000010      MOV    #COLOPN,R0      ;Get operand column number
29 010534 004737 012064'    CALL    ODCTAB      ;Tab over to operand field
30     ;
31     ; Display the floating point operand
32     ;
33 010540 010200      MOV    R2,R0      ;Get operand code
34 010542 004737 011672'    CALL    ODCFPU      ;Display general FPU operand
35     ;
36     ; Finished
37     ;
38 010546 000207      RETURN
39     ; -----
40     ;
41     ; TYP13 -- Floating accumulator and floating source operand.
42     ;
43 010550 ODC13:
44     ;
45     ; Tab over to the operand field
46     ;
47 010550 012700 000010      MOV    #COLOPN,R0      ;Get operand column number
48 010554 004737 012064'    CALL    ODCTAB      ;Tab over to operand field
49     ;
50     ; Display the source operand
51     ;
52 010560 010200      MOV    R2,R0      ;Get instruction word
53 010562 004737 011672'    CALL    ODCFPU      ;Display the source operand
54 010566      TTYOUT #COMMA      ;Display comma
55 010576 005237 000000G    INC     D.PCOL      ;Advance column number
56     ;
57     ; Display the accumulator number

```

ODCxxx -- Display instruction operands

```

58
59 010602 010200          ;      MOV      R2,R0          ;Get accumulator number
60 010604 072027 177772   ;      ASH      #-6,R0       ;Right justify
61 010610 004737 011620'   ;      CALL     ODCACC       ;Display the accumulator number
62                          ;
63                          ; Finished
64                          ;
65 010614 000207          ;      RETURN

```

```

1      ;-----
2      ;   TYP14 -- Floating accumulator and general source operand.
3      ;
4      010616      ODC14:
5      ;
6      ;   Tab over to the operand field
7      ;
8      010616      012700      000010      MOV      #COLOPN,RO      ;Get operand column number
9      010622      004737      012064'      CALL      ODCTAB      ;Tab over to operand field
10     ;
11     ;   Display the source operand
12     ;
13     010626      010200      MOV      R2,RO      ;Get instruction word
14     010630      004737      011000'      CALL      ODCGEN      ;Display the source operand
15     010634      .TTYOUT      #COMMA      ;Display comma
16     010644      005237      00000000      INC      D.PCOL      ;Advance column number
17     ;
18     ;   Display the accumulator number
19     ;
20     010650      010200      MOV      R2,RO      ;Get accumulator number
21     010652      072027      177772      ASH      #-6,RO      ;Right justify
22     010656      004737      011620'      CALL      ODCACC      ;Display the accumulator number
23     ;
24     ;   Finished
25     ;
26     010662      000207      RETURN

```

1					
2					
3					
4	010664				
5					
6					
7					
8	010664	012700	000010		
9	010670	004737	012064'		
10					
11					
12					
13	010674	010200			
14	010676	072027	177772		
15	010702	004737	011620'		
16	010706				
17	010716	005237	0000000		
18					
19					
20					
21	010722	010200			
22	010724	004737	011672'		
23					
24					
25					
26	010730	000207			

```

;-----
;  TYP15 -- Floating accumulator and floating destination operand.
;
; ODC15:
;
;  Tab over to the operand field
;
;      MOV      #COLOPN,RO      ;Get operand column number
;      CALL     ODCTAB          ;Tab over to operand field
;
;  Display the floating accumulator
;
;      MOV      R2,RO           ;Get the accumulator number
;      ASH      #-6,RO          ;Right justify
;      CALL     ODCACC          ;Display the accumulator
;      .TTYOUT  #COMMA          ;Display comma
;      INC      D.PCOL          ;Advance column number
;
;  Display the floating destination
;
;      MOV      R2,RO           ;Get the destination operand code
;      CALL     ODCFPU          ;Display it
;
;  Finished
;
;      RETURN

```

ODCxxx -- Display instruction operands

```

1      ; -----
2      ;   TYP16 -- Floating accumulator and general destination operand.
3      ;
4 010732 ODC16:
5      ;
6      ;   Tab over to the operand field
7      ;
8 010732 012700 000010      MOV      #COLOPN,R0      ;Get operand column number
9 010736 004737 012064'    CALL      ODC16TAB      ;Tab over to operand field
10     ;
11     ;   Display the floating accumulator
12     ;
13 010742 010200      MOV      R2,R0      ;Get the accumulator number
14 010744 072027 177772    ASH      #-6,R0      ;Right justify
15 010750 004737 011620'    CALL      ODCACC      ;Display the accumulator
16 010754      .TTYOUT #COMMA      ;Display comma
17 010764 005237 000000G    INC      D.PCOL      ;Advance column number
18     ;
19     ;   Display the destination
20     ;
21 010770 010200      MOV      R2,R0      ;Get the destination operand code
22 010772 004737 011000'    CALL      ODCGEN      ;Display it
23     ;
24     ;   Finished
25     ;
26 010776 000207      RETURN

```

ODCGEN -- Display general operand value

```

1          .SBTTL  ODCGEN -- Display general operand value
2          -----
3          ; ODCGEN is called during instruction decoding to display a general
4          ; operand of any mode.
5          ;
6          ; Inputs:
7          ;   R0 = Operand value from instruction.
8          ;   R1 = Address of word following instruction.
9          ;
10         ; Outputs:
11         ;   R1 is incremented if the addressing mode uses a value in memory that
12         ;   follows the instruction.
13         ;
14 011000 010246 ODCGEN: MOV     R2, -(SP)
15 011002 010346      MOV     R3, -(SP)
16
17         ; Get the addressing mode
18
19 011004 010002      MOV     R0, R2
20 011006 072227 177776      ASH     #-2, R2      ; Right justify 2* mode
21 011012 042702 177761      BIC     #^C16, R2    ; Mask out all but 2* mode
22
23         ; Determine if the register is R7 or is in the range R0 to R6.
24
25 011016 010003      MOV     R0, R3      ; Get mode and register value
26 011020 042703 177770      BIC     #^C7, R3    ; Mask out all but register number
27 011024 020327 000007      CMP     R3, #7      ; Is register R7?
28 011030 001003      BNE     1$
29
30         ; Addressing mode is relative to R7.
31         ; Call display routine based on the addressing mode.
32         ; R0 and R3 contain the mode and register number.
33
34 011032 004772 011452'      CALL    @AD7VEC(R2) ; Call the processing routine
35 011036 000402              BR      9$
36
37         ; Jump to processing routine based on the mode
38
39 011040 004772 011432' 1$: CALL    @ADRVEC(R2) ; Jump to processing routine
40
41         ; Finished
42
43 011044 012603 9$:  MOV     (SP)+, R3
44 011046 012602      MOV     (SP)+, R2
45 011050 000207      RETURN

```

```

1      ; -----
2      ;   Display routines for registers R0 through R6
3      ;
4      ;   Mode 0 -- R
5      ;
6 011052 004737 011472' ADRMD0: CALL   ODCREG      ; Display register
7 011056 000207          RETURN
8      ;
9      ;   Mode 1 -- (R)
10     ;
11 011060 010046 ADRMD1: MOV     RO,-(SP)      ; Save the register number
12 011062          .TTYOUT #'('              ; Open paren
13 011072 012600          MOV     (SP)+,RO    ; Recover the register number
14 011074 004737 011472' CALL   ODCREG      ; Display the register number
15 011100          .TTYOUT #')              ; Close paren
16 011110 000207          RETURN
17     ;
18     ;   Mode 2 -- (R)+
19     ;
20 011112 004737 011060' ADRMD2: CALL   ADRMD1      ; Print "(r)"
21 011116          .TTYOUT #'+'              ; Put in trailing plus sign
22 011126 000207          RETURN
23     ;
24     ;   Mode 3 -- @(R)+
25     ;
26 011130 010046 ADRMD3: MOV     RO,-(SP)      ; Save the register number
27 011132          .TTYOUT #'@              ; Print "@"
28 011142 012600          MOV     (SP)+,RO    ; Recover the register number
29 011144 004737 011112' CALL   ADRMD2      ; Print "(r)+"
30 011150 000207          RETURN
31     ;
32     ;   Mode 4 -- -(R)
33     ;
34 011152 010046 ADRMD4: MOV     RO,-(SP)      ; Save the register number
35 011154          .TTYOUT #'-'              ; Print "-"
36 011164 012600          MOV     (SP)+,RO    ; Recover the register number
37 011166 004737 011060' CALL   ADRMD1      ; Print "(r)"
38 011172 000207          RETURN
39     ;
40     ;   Mode 5 -- @-(R)
41     ;
42 011174 010046 ADRMD5: MOV     RO,-(SP)      ; Save the register number
43 011176          .TTYOUT #'@              ; Print "@-"
44 011206          .TTYOUT #'-'              ;
45 011216 012600          MOV     (SP)+,RO    ; Recover register number
46 011220 004737 011060' CALL   ADRMD1      ; Print "(r)"
47 011224 000207          RETURN
48     ;
49     ;   Mode 6 -- X(R)
50     ;
51 011226 010046 ADRMD6: MOV     RO,-(SP)      ; Save the register number
52 011230 106521 MFPD     (R1)+              ; Get the vector base address
53 011232 062737 000002 000000G ADD     #2,D,ILEN      ; Say instruction uses 2 more bytes
54 011240 012600          MOV     (SP)+,RO    ; Get value to RO
55 011242 004737 011720' CALL   ODCADR      ; Display the address
56 011246 012600          MOV     (SP)+,RO    ; Recover the register number
57 011250 004737 011060' CALL   ADRMD1      ; Print "(r)"

```

58 011254 000207

RETURN

59

;

60

; Mode 7 -- @X(R)

61

;

62 011256 010046

ADRMD7: MOV RO, -(SP)

; Save register number

63 011260

. TTYOUT #'@

; Print "@"

64 011270 012600

MOV (SP)+, RO

; Recover register number

65 011272 004737 011226'

CALL ADRMD6

; Print "x(r)"

66 011276 000207

RETURN

ODCGEN -- Display general operand value

```

1      ; -----
2      ;   Display routines for the various modes with R7.
3      ;
4      ;   Mode 2 -- #n
5      ;
6 011300 AD7MD2: .TTYOUT #'#           ;Print "#"
7 011310      MFPD      (R1)+         ;Get the value onto the stack
8 011312      ADD       #2,D. ILEN    ;Say instruction uses 2 more bytes
9 011320      MOV       (SP)+,R0      ;Get the value
10 011322      CALL      ODCNEG        ;See if value should be printed as neg number
11 011326      BCC       9$           ;Br if it was a negative number
12 011330      CALL      PRTOCT        ;Print as an octal value
13 011334      000207
14      9$:      RETURN
15      ;
16      ;   Mode 3 -- @#address
17      ;
18 011336 AD7MD3: .TTYOUT #'@         ;Print "@#"
19 011346      .TTYOUT #'#
20 011356      MFPD      (R1)+         ;Get the address
21 011360      ADD       #2,D. ILEN    ;Advance the instruction length
22 011366      MOV       (SP)+,R0      ;Get the address
23 011370      CALL      ODCADR        ;Display the address
24 011374      000207
25      ;
26      ;   Mode 6 -- address
27      ;
28 011376 AD7MD6: MFPD      (R1)+         ;Get the address
29 011400      ADD       #2,D. ILEN    ;Advance the instruction length
30 011406      MOV       (SP)+,R0      ;Get the address
31 011410      ADD       R1,R0         ;Relocate the address
32 011412      CALL      ODCADR        ;Display the address
33 011416      000207
34      ;
35      ;   Mode 7 -- @address
36      ;
37 011420 AD7MD7: .TTYOUT #'@         ;Print "@"
38 011430      BR        AD7MD6        ;Go display the address

```

```
1 ; -----  
2 ; Jump table used to dispatch to the correct display routine for  
3 ; addressing modes using registers R0 through R6.  
4 ;  
5 011432 011052' ADRVEC: .WORD ADRMD0 ; Mode 0  
6 011434 011060' .WORD ADRMD1 ; Mode 1  
7 011436 011112' .WORD ADRMD2 ; Mode 2  
8 011440 011130' .WORD ADRMD3 ; Mode 3  
9 011442 011152' .WORD ADRMD4 ; Mode 4  
10 011444 011174' .WORD ADRMD5 ; Mode 5  
11 011446 011226' .WORD ADRMD6 ; Mode 6  
12 011450 011256' .WORD ADRMD7 ; Mode 7  
13 ;  
14 ; Jump table used to dispatch to the correct display routine for  
15 ; addressing modes using register R7 (PC).  
16 ;  
17 011452 011052' AD7VEC: .WORD ADRMD0 ; Mode 0  
18 011454 011060' .WORD ADRMD1 ; Mode 1  
19 011456 011300' .WORD AD7MD2 ; Mode 2  
20 011460 011336' .WORD AD7MD3 ; Mode 3  
21 011462 011152' .WORD ADRMD4 ; Mode 4  
22 011464 011174' .WORD ADRMD5 ; Mode 5  
23 011466 011376' .WORD AD7MD6 ; Mode 6  
24 011470 011420' .WORD AD7MD7 ; Mode 7
```

```

1          .SBTTL  ODCREG -- Display register name
2          ;-----
3          ; ODCREG is called to display the name of a register.
4          ;
5          ; Inputs:
6          ;   RO = Register number
7          ;
8 011472    010046  ODCREG: MOV      RO,-(SP)
9          ;
10         ; Determine which register this is
11         ;
12 011474    042700  177770      BIC      #^C7,RO      ; Clear all but the register number
13         ;
14         ; See if this is R7 (PC)
15         ;
16 011500    020027  000007      CMP      RO,#7        ; Is register PC?
17 011504    001011      BNE      1$
18 011506      .TTYOUT #'P        ; Display "PC"
19 011516      .TTYOUT #'C
20 011526    000427      BR       9$
21         ;
22         ; See if this is R6 (SP)
23         ;
24 011530    020027  000006  1$:      CMP      RO,#6        ; Is register SP?
25 011534    001011      BNE      2$        ; Br if not
26 011536      .TTYOUT #'S        ; Print "SP"
27 011546      .TTYOUT #'P
28 011556    000413      BR       9$
29         ;
30         ; This is a register in the range RO to R5.
31         ;
32 011560      2$:      .TTYOUT #'R        ; Print "Rn"
33 011570    011600      MOV      (SP),RO      ; Get register number
34 011572    042700  177770      BIC      #^C7,RO
35 011576    062700  000060      ADD      #'0,RO      ; Convert to ascii digit
36 011602      .TTYOUT          ; Print it
37         ;
38         ; Finished
39         ;
40 011606    062737  000002  000000G 9$:      ADD      #2,D.PCOL      ; Advance print column counter
41 011614    012600      MOV      (SP)+,RO
42 011616    000207      RETURN

```

ODCACC -- Print a floating point accumulator name

```

1          .SBTTL  ODCACC -- Print a floating point accumulator name
2          ; -----
3          ; Print the name of a floating point accumulator.
4          ;
5          ; Inputs:
6          ;   RO = Accumulator number
7          ;
8 011620 010046 ODCACC: MOV      RO, -(SP)
9          ;
10         ; Print ACn
11         ;
12 011622         .TTYOUT #'A
13 011632         .TTYOUT #'C
14 011642 011600 MOV      (SP), RO
15 011644 042700 177774 BIC      #^C3, RO      ; Clear all but ACC #
16 011650 062700 000060 ADD      #'O, RO      ; Convert to ascii char
17 011654         .TTYOUT         ; Print the #
18         ;
19         ; Finished
20         ;
21 011660 062737 000003 000000G ADD      #3, D.PCOL      ; Advance print column number
22 011666 012600 MOV      (SP)+, RO
23 011670 000207 RETURN

```

ODCFPU -- Print general floating point operand

```

1          .SBTTL  ODCFPU -- Print general floating point operand
2          ;-----
3          ; ODCFPU is called to display a general floating point operand.
4          ;
5          ; Inputs:
6          ;   RO = Floating point operand mode and register #.
7          ;
8 011672 010046 ODCFPU: MOV      RO,-(SP)
9          ;
10         ; Determine if it is a floating point accumulator
11         ;
12 011674 032700 000070          BIT      #70,RO          ; Is mode = 0 (accumulator)?
13 011700 001003          BNE      1$          ; Br if not
14 011702 004737 011620'          CALL    ODCACC          ; Display an accumulator name
15 011706 000402          BR       9$
16         ;
17         ; It is not an accumulator
18         ;
19 011710 004737 011000' 1$:      CALL    ODCGEN          ; Display general operand
20         ;
21         ; Finished
22         ;
23 011714 012600 9$:      MOV      (SP)+,RO
24 011716 000207          RETURN

```

ODCADR -- Display decoded instruction address

```

1          .SBTTL  ODCADR -- Display decoded instruction address
2          ;-----
3          ; ODCADR is called to display an address value which is an instruction
4          ; operand.  If the address is relative to a relocation base, the
5          ; address is displayed in the form "[r,offset]".
6          ; If the address is not relative to a relocation base, it is displayed
7          ; in the form "address".
8          ;
9          ; Inputs:
10         ;   RO = Address to be displayed.
11         ;
12 ODCADR: MOV     RO, -(SP)
13         MOV     R3, -(SP)
14         ;
15         ; Determine if address should be shown as a negative number
16         ;
17 011724 004737 012022'      CALL     ODCNEG      ;Should this be shown as a negative number?
18 011730 103031              BCC      9$          ;Br if yes
19         ;
20         ; Determine if address is relative to a relocation base
21         ;
22 011732 032737 000000G 000000G      BIT      #DP$DAA, D, PFMT ;Should address be absolute?
23 011740 001023              BNE      1$          ;Br if yes
24 011742 004737 007412'      CALL     RELADR      ;See if address is within relocation region
25 011746 103420              BCS      1$          ;Br if address is not within relocation region
26         ;
27         ; Address is within a relocation region.
28         ; Display in the form "[r,address]".
29         ;
30         MOV     RO, R3          ;Save the address
31         .TTYOUT #'[            ;Print "[r,address]"
32         MOV     R3, RO          ;Get back address
33 011764 004737 007322'      CALL     SHOADR      ;Display the address
34 011770              .TTYOUT #']'      ;Put in closing bracket
35 012000 062737 000002 000000G      ADD     #2, D, PCOL      ;Advance print column counter
36 012006 000402              BR       9$
37         ;
38         ; Address is not relative to a relocation base
39         ;
40 012010 004737 006030'      1$:      CALL     PRTOCT      ;Display the address
41         ;
42         ; Finished
43         ;
44 012014 012603      9$:      MOV     (SP)+, R3
45 012016 012600              MOV     (SP)+, RO
46 012020 000207              RETURN

```

```

1          .SBTTL  ODCNEG -- Determine if values should be shown as negative
2          ;-----
3          ; ODCNEG is called to determine if an address or value should be displayed
4          ; as a negative number.
5          ; Currently, the only values that are shown as negative numbers
6          ; are -1 and -2.
7          ;
8          ; Inputs:
9          ; RO = Value to be tested.
10         ;
11         ; Outputs:
12         ; C-flag cleared ==> Value was displayed as a negative number.
13         ; C-flag set    ==> Value is not displayed as a negative number.
14         ;
15 012022  ODCNEG:
16         ;
17         ; See if this is a value to be displayed as a negative number
18         ;
19 012022  020027 177776      CMP     RO,#177776      ;Is this a negative value?
20 012026  103414          BLO     9$                ;Br if not
21         ;
22         ; This is a negative value
23         ;
24 012030  010046          MOV     RO,-(SP)
25 012032          .TTYOUT #'-'                ;Print minus sign
26 012042  011600          MOV     (SP),RO        ;Get the value
27 012044  005400          NEG     RO              ;Negate it
28 012046  004737 006030'  CALL    PRTOCT        ;Print it
29 012052  012600          MOV     (SP)+,RO       ;Recover original value
30 012054  000241          CLC                    ;This is a negative number
31 012056  000401          BR      10$
32         ;
33         ; Finished
34         ;
35 012060  000261          9$:      SEC                    ;This is not a negative value
36 012062  000207          10$:     RETURN

```

ODCTAB -- Tab to a specified column

```

1
2
3
4
5
6
7
8
9
10
11
12 012064 010046
13
14
15
16 012066
17 012076 005237 000000G
18 012102 023716 000000G
19 012106 103767
20
21
22
23 012110 012600
24 012112 000207

```

```

.SBTTL ODCTAB -- Tab to a specified column
-----
; ODCTAB is called to tab over to a specified print column.
;
; Inputs:
;   D.PCOL = Current print column number.
;   RO     = Desired print column.
;
; Outputs:
;   D.PCOL = New print column after tabbing.
;
ODCTAB: MOV     RO, -(SP)
;
; Print spaces until we reach the desired column
;
1$:      .TTYOUT #SPACE      ;Print a space
        INC      D.PCOL      ;Advance column number
        CMP      D.PCOL, (SP) ;Reached desired column?
        BLO      1$          ;Loop if not
;
; Finished
;
        MOV      (SP)+, RO
        RETURN

```

```

1      .SBTTL  INSBAS -- Instruction decoding tables
2
3      ;-----
4      ; Table used for decoding a PDP-11 instruction into symbolic form.
5      ; Each instruction is defined by an invocation of the INSTR macro
6      ; which takes 4 arguments. The four arguments are:
7      ;   Argument 1 = 16 bit mask used to mask out the operand portions
8      ;                 of the instruction word.
9      ;   Argument 2 = Value of instruction after mask (arg 1) has been applied.
10     ;   Argument 3 = Operand type code for the instruction.
11     ;   Argument 4 = Symbolic name for the instruction.
12
13     ; Define symbolic names for offsets into an instruction description block:
14     ;
15     ; IB$MSK = 0 ; Mask value
16     ; IB$VAL = 2 ; Instruction value
17     ; IB$TYP = 4 ; Operand type
18     ; IB$LEN = 5 ; Length of instruction mnemonic text string
19     ; IB$NAM = 6 ; Start of instruction mnemonic text string
20
21     .MACRO  INSTR  MASK, VALUE, TYPE, NAME
22     .WORD  MASK
23     .WORD  VALUE
24     .BYTE  TYPE
25     .NCHR  NAMLEN, NAME
26     .BYTE  NAMLEN
27     .ASCII  '/'NAME/'
28     .EVEN
29     .ENDM  INSTR
30
31     ; Type codes for the instruction operand types:
32     ; TYP1 = No operand.
33     ; TYP2 = Single general operand.
34     ; TYP4 = 4-bit absolute numeric operand.
35     ; TYP5 = Condition code modification instruction.
36     ; TYP6 = Branch offset operand.
37     ; TYP7 = Register source, general operand destination.
38     ; TYP9 = General source and destination operands.
39     ; TYP10 = Register and loop offset (SOB).
40     ; TYP11 = EMT and TRAP instructions.
41     ; TYP12 = Single floating point general operand
42     ; TYP13 = Floating point accumulator and floating source operand
43     ; TYP14 = Floating point accumulator and general source operand
44     ; TYP15 = Floating point accumulator and floating destination operand
45     ; TYP16 = Floating point accumulator and general destination operand
46
47     ; Define the instructions
48
49     INSBAS:
50     ;
51     ; Mask Code Type Name
52     ;-----
53     INSTR 000000, 000000, TYP1, HALT
54     INSTR 000000, 000001, TYP1, WAIT
55     INSTR 000000, 000002, TYP1, RTI
56     INSTR 000000, 000003, TYP1, BPT
57     INSTR 000000, 000004, TYP1, IOT
58     INSTR 000000, 000005, TYP1, RESET
59     INSTR 000000, 000006, TYP1, RTT

```

INSBAS -- Instruction decoding tables

58 012224	INSTR	000077,	000100,	TYP2,	JMP
59 012236	INSTR	000000,	000207,	TYP1,	RETURN
60 012252	INSTR	000007,	000200,	TYP3,	RTS
61 012264	INSTR	000007,	000230,	TYP4,	SPL
62 012276	INSTR	000000,	000240,	TYP1,	NOP
63 012310	INSTR	000017,	000240,	TYP5,	CL
64 012320	INSTR	000017,	000260,	TYP5,	SE
65 012330	INSTR	000077,	000300,	TYP2,	SWAB
66 012342	INSTR	000377,	000400,	TYP6,	BR
67 012352	INSTR	000377,	001000,	TYP6,	BNE
68 012364	INSTR	000377,	001400,	TYP6,	BEQ
69 012376	INSTR	000377,	002000,	TYP6,	BGE
70 012410	INSTR	000377,	002400,	TYP6,	BLT
71 012422	INSTR	000377,	003000,	TYP6,	BGT
72 012434	INSTR	000377,	003400,	TYP6,	BLE
73 012446	INSTR	000077,	004700,	TYP2,	CALL
74 012460	INSTR	000777,	004000,	TYP7,	JSR
75 012472	INSTR	000077,	005000,	TYP2,	CLR
76 012504	INSTR	000077,	005100,	TYP2,	COM
77 012516	INSTR	000077,	005200,	TYP2,	INC
78 012530	INSTR	000077,	005300,	TYP2,	DEC
79 012542	INSTR	000077,	005400,	TYP2,	NEG
80 012554	INSTR	000077,	005500,	TYP2,	ADC
81 012566	INSTR	000077,	005600,	TYP2,	SBC
82 012600	INSTR	000077,	005700,	TYP2,	TST
83 012612	INSTR	000077,	006000,	TYP2,	ROR
84 012624	INSTR	000077,	006100,	TYP2,	ROL
85 012636	INSTR	000077,	006200,	TYP2,	ASR
86 012650	INSTR	000077,	006300,	TYP2,	ASL
87 012662	INSTR	000077,	006400,	TYP8,	MARK
88 012674	INSTR	000077,	006500,	TYP2,	MFPI
89 012706	INSTR	000077,	006600,	TYP2,	MTPI
90 012720	INSTR	000077,	006700,	TYP2,	SXT
91 012732	INSTR	007777,	010000,	TYP9,	MOV
92 012744	INSTR	007777,	020000,	TYP9,	CMP
93 012756	INSTR	007777,	030000,	TYP9,	BIT
94 012770	INSTR	007777,	040000,	TYP9,	BIC
95 013002	INSTR	007777,	050000,	TYP9,	BIS
96 013014	INSTR	007777,	060000,	TYP9,	ADD
97 013026	INSTR	000777,	070000,	TYP8,	MUL
98 013040	INSTR	000777,	071000,	TYP8,	DIV
99 013052	INSTR	000777,	072000,	TYP8,	ASH
100 013064	INSTR	000777,	073000,	TYP8,	ASHC
101 013076	INSTR	000777,	074000,	TYP7,	XOR
102 013110	INSTR	000007,	075000,	TYP1,	FADD
103 013122	INSTR	000007,	075010,	TYP1,	FSUB
104 013134	INSTR	000007,	075020,	TYP1,	FMUL
105 013146	INSTR	000007,	075030,	TYP1,	FDIV
106 013160	INSTR	000777,	077000,	TYP10,	SOB
107 013172	INSTR	000377,	100000,	TYP6,	BPL
108 013204	INSTR	000377,	100400,	TYP6,	BMI
109 013216	INSTR	000377,	101000,	TYP6,	BHI
110 013230	INSTR	000377,	101400,	TYP6,	BLOS
111 013242	INSTR	000377,	102000,	TYP6,	BVC
112 013254	INSTR	000377,	102400,	TYP6,	BVS
113 013266	INSTR	000377,	103000,	TYP6,	BCC
114 013300	INSTR	000377,	103400,	TYP6,	BCS

INSBAS -- Instruction decoding tables

115 013312	INSTR	000377,	104000,	TYP11,	EMT
116 013324	INSTR	000377,	104400,	TYP11,	TRAP
117 013336	INSTR	000077,	105000,	TYP2,	CLRB
118 013350	INSTR	000077,	105100,	TYP2,	COMB
119 013362	INSTR	000077,	105200,	TYP2,	INCB
120 013374	INSTR	000077,	105300,	TYP2,	DECB
121 013406	INSTR	000077,	105400,	TYP2,	NEGB
122 013420	INSTR	000077,	105500,	TYP2,	ADCB
123 013432	INSTR	000077,	105600,	TYP2,	SBCB
124 013444	INSTR	000077,	105700,	TYP2,	TSTB
125 013456	INSTR	000077,	106000,	TYP2,	RORB
126 013470	INSTR	000077,	106100,	TYP2,	ROLB
127 013502	INSTR	000077,	106200,	TYP2,	ASRB
128 013514	INSTR	000077,	106300,	TYP2,	ASLB
129 013526	INSTR	000077,	106500,	TYP2,	MFPD
130 013540	INSTR	000077,	106600,	TYP2,	MTPD
131 013552	INSTR	007777,	110000,	TYP9,	MOVB
132 013564	INSTR	007777,	120000,	TYP9,	CMPB
133 013576	INSTR	007777,	130000,	TYP9,	BITB
134 013610	INSTR	007777,	140000,	TYP9,	BICB
135 013622	INSTR	007777,	150000,	TYP9,	BISB
136 013634	INSTR	007777,	160000,	TYP9,	SUB
137 013646	INSTR	000000,	170000,	TYP1,	CFCC
138 013660	INSTR	000000,	170001,	TYP1,	SETF
139 013672	INSTR	000000,	170002,	TYP1,	SETI
140 013704	INSTR	000000,	170011,	TYP1,	SETD
141 013716	INSTR	000000,	170012,	TYP1,	SETL
142 013730	INSTR	000077,	170100,	TYP2,	LDFPS
143 013744	INSTR	000077,	170200,	TYP2,	STFPS
144 013760	INSTR	000077,	170300,	TYP2,	STST
145 013772	INSTR	000077,	170400,	TYP12,	CLRF
146 014004	INSTR	000077,	170500,	TYP12,	TSTF
147 014016	INSTR	000077,	170600,	TYP12,	ABSF
148 014030	INSTR	000077,	170700,	TYP12,	NEGF
149 014042	INSTR	000377,	171000,	TYP13,	MULF
150 014054	INSTR	000377,	171400,	TYP13,	MODF
151 014066	INSTR	000377,	172000,	TYP13,	ADDF
152 014100	INSTR	000377,	172400,	TYP13,	LDF
153 014112	INSTR	000377,	173000,	TYP13,	SUBF
154 014124	INSTR	000377,	173400,	TYP13,	CMPF
155 014136	INSTR	000377,	174000,	TYP15,	STF
156 014150	INSTR	000377,	174400,	TYP13,	DIVF
157 014162	INSTR	000377,	175000,	TYP16,	STEXP
158 014176	INSTR	000377,	175400,	TYP16,	STCFI
159 014212	INSTR	000377,	176000,	TYP16,	STCFD
160 014226	INSTR	000377,	176400,	TYP14,	LDEXP
161 014242	INSTR	000377,	177000,	TYP14,	LDCIF
162 014256	INSTR	000377,	177400,	TYP13,	LDCDF
163 014272	INSTR	000077,	106400,	TYP2,	MTPS
164 014304	INSTR	000077,	106700,	TYP2,	MFPS
165 014316	INSTR	177777,	000000,	TYP1,	????

INSEND:

;

; Address vector used to jump to correct routine to decode and display
 ; the operands associated with an instruction.

;

. MACRO TYPDEF TYPNAM,RTN

INSBAS -- Instruction decoding tables

```

172          TYPNAM =      .-TYPVEC
173          .WORD      RTN
174          .ENDM      TYPDEF
175          ;
176          ; Define the types
177          ;
178          TYPVEC:
179          TYPDEF      TYP1,ODC1      ;No operand
180          TYPDEF      TYP2,ODC2      ;Single general operand.
181          TYPDEF      TYP3,ODC3      ;Single register operand (RTS)
182          TYPDEF      TYP4,ODC4      ;4-bit absolute numeric operand.
183          TYPDEF      TYP5,ODC5      ;Condition code modification instruction.
184          TYPDEF      TYP6,ODC6      ;Branch offset operand.
185          TYPDEF      TYP7,ODC7      ;Register source, general operand destination.
186          TYPDEF      TYP8,ODC8      ;Register destination, general source operand
187          TYPDEF      TYP9,ODC9      ;General source and destination operands.
188          TYPDEF      TYP10,ODC10     ;Register and loop offset (SOB).
189          TYPDEF      TYP11,ODC11     ;EMT and TRAP instructions.
190          TYPDEF      TYP12,ODC12     ;One floating point operand.
191          TYPDEF      TYP13,ODC13     ;Floating accum and floating source operand.
192          TYPDEF      TYP14,ODC14     ;Floating accum and general source operand.
193          TYPDEF      TYP15,ODC15     ;Floating accumulator and floating destination
194          TYPDEF      TYP16,ODC16     ;Floating accumulator and general destination.
195
196          000001      .END

```

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 9608 Words (38 Pages)
 Size of core pool: 18176 Words (71 Pages)
 Operating system: RT-11

Elapsed time: 00:01:40.43
 ,LP:TSDEBUG=DK:TSDEBUG/C/N:SYM

\$DBGBK	1-33	6-16										
\$DEBUG	1-34	6-17										
... V1	10-32	20-27	22-5	32-22	32-27	33-17	33-42	34-26	35-22	39-49	39-52	39-59
	41-27	45-55	47-29	47-32	47-35	47-38	48-41	49-15	50-16	51-16	52-54	53-15
	54-16	55-16	57-12	57-15	57-21	57-27	57-35	57-43	57-44	57-63	58-6	58-17
	58-18	58-36	60-18	60-19	60-26	60-27	60-32	60-36	61-12	61-13	61-17	63-31
	63-34	64-25	65-16									
ACRNUM	29-42	29-54	30-10#									
ACRVAL	9-20	9-31	29-15#									
AD7MD2	58-6#	59-19										
AD7MD3	58-17#	59-20										
AD7MD6	58-27#	58-37	59-23									
AD7MD7	58-36#	59-24										
AD7VEC	56-34	59-17#										
ADRMD0	57-6#	59-5	59-17									
ADRMD1	57-11#	57-20	57-37	57-46	57-57	59-6	59-18					
ADRMD2	57-20#	57-29	59-7									
ADRMD3	57-26#	59-8										
ADRMD4	57-34#	59-9	59-21									
ADRMD5	57-42#	59-10	59-22									
ADRMD6	57-51#	57-65	59-11									
ADRMD7	57-62#	59-12										
ADRVEC	56-39	59-5#										
BELL	1-72#											
BKSPAC	1-73#	2-22										
BRKCHK	7-13	26-22#										
BRKENT	1-21	6-29	7-8#									
BRKRST	7-67	24-7#										
BRKSET	7-24	8-14	25-6#									
CHKADR	15-20	44-29#										
CMDAT	2-31	12-6#										
CMDAT1	12-7#	12-35										
CMDB	2-36	15-5#										
CMDBK1	10-22#	12-26	20-35	22-30								
CMDBKS	2-30	10-8	10-21#	11-17	21-22							
CMDBSP	2-47	22-5#										
CMDCR	2-44	20-5#										
CMDDCD	2-34	11-5#	21-8	21-17								
CMDEXP	2-33	13-5#										
CMDG	2-37	16-5#										
CMDI	2-38	16-39#										
CMDLF	2-45	18-58	20-17#									
CMDM	2-39	19-5#										
CMDP	2-40	16-16#	17-31									
CMDR	2-41	14-5#										
CMDRSB	2-35	21-5#										
CMDS	2-42	16-26#										
CMDSL1	10-7#	11-20	12-20	20-30	22-25							
CMDSLH	2-29	10-6#										
CMDTAB	2-48	17-9#										
CMDTBL	2-4#	2-24	9-39									
CMDUP	2-46	22-6	22-11#									
CMDUS	2-32	12-32#										
CMDVEC	2-29#	9-49										
CMDX	2-43	18-5#										
COLOPN	1-81#	46-14	46-33	47-8	48-8	48-33	49-8	50-8	51-8	52-8	52-28	52-47

	53-8	54-8	55-8									
COMMA	1-76#	48-41	49-15	50-16	51-16	52-54	53-15	54-16	55-16			
CORUSR	1-32	6-15	43-15									
CR	1-70#	1-92	1-94	2-19	18-15	18-25	18-40	20-27	43-29			
CRCHAR	1-94#	20-32	21-19									
CRLF	1-93#	5-31	5-39	9-5	12-9	16-8	16-19	17-65	21-14	22-22	22-27	27-7
	27-28	27-41	27-56	27-68	35-6							
D\$BKST	1-30	24-8	24-26	25-21								
D\$CKBK	1-33	17-60	24-10	25-7	26-53	28-19	28-24					
D\$DBRK	1-28	5-43	7-14	26-26	26-49	27-11						
D\$DMON	1-29	7-22	7-65	8-6	8-12	8-28	8-54	19-5	19-8	26-30		
D\$DVAL	1-30	19-5	19-17	26-37								
D\$FBRK	1-31	6-28	7-14	7-77	8-5	27-54						
D\$IBRK	1-28	5-43	7-14	7-68	26-26	26-85	27-39					
D\$INIT	1-34	6-22	23-16									
D\$IPND	1-30	7-21	7-64	7-71	8-8	8-12	8-54	26-59				
D\$ISPC	1-36	16-39	16-44	37-33	37-45	37-58	38-28	38-40				
D\$RUN	1-33	7-39	8-10									
D\$SBRK	1-28	5-43	7-14	26-26	26-66	27-32						
D\$SSTP	1-28	7-22	7-65	8-6	8-12	8-54	16-26	16-31	17-64	26-53	26-64	
D\$TSTP	1-34	7-22	7-65	7-84	8-6	8-12	8-54	17-30	26-53	26-64		
D. BKAD	1-25	3-33	15-13	15-21*	24-16	25-13	26-75	28-12				
D. BKNM	1-28	26-67*	26-84*	27-42	27-47							
D. BKSV	1-29	24-21	25-16*									
D. BYTM	1-25	10-9*	10-22*	20-24	21-11	22-19	37-26	37-67	38-22	38-54		
D. CBRK	1-35	17-59*	27-44*									
D. DADR	1-29	8-30	19-10*	26-32	27-17	27-20						
D. DOLD	1-29	8-35*	26-40	27-23*	27-27*							
D. DTRG	1-29	19-18*	26-44									
D. END	1-31	23-13										
D. FLAG	1-28	5-43*	6-22	6-28*	7-14	7-21*	7-22	7-39*	7-64*	7-65	7-68	7-71*
	7-77	7-84*	8-5*	8-6	8-8*	8-10*	8-12	8-28	8-54	16-26*	16-31*	16-39*
	16-44*	17-30*	17-60*	17-64*	19-5*	19-8*	19-17*	23-16*	24-8	24-10	24-26*	25-7
	25-21*	26-26*	26-30	26-37	26-49*	26-53	26-59	26-64	26-66*	26-85*	27-11	27-32
	27-39	27-54	28-19*	28-24*	37-33	37-45	37-58	38-28	38-40			
D. ILEN	1-34	10-10*	10-23*	18-9*	21-13	45-20*	57-53*	58-8*	58-20*	58-28*		
D. LOC	1-25	5-54*	8-9*	10-7	11-12	12-7*	12-33	13-22	13-29	20-10*	20-18	20-26*
	20-31*	21-9	21-13*	21-18*	22-12	22-21*	22-26*	36-16*	37-32	37-43	37-50	37-72*
	37-78	37-80	38-27	38-38	38-46	38-59	38-64	38-67	39-23	39-29		
D. LOCM	1-26	11-18	12-8*	36-15*	37-16	37-21	38-17	39-18				
D. LVAL	1-33	10-12*	10-25*	12-6	12-32	14-36	14-51	18-6				
D. MASK	1-33	3-31	8-32	23-15*	26-34	27-25						
D. NMBE	1-31	30-38										
D. NMBF	1-25	30-28	30-54									
D. PCNT	1-32	7-75	7-79*	16-9*	16-18*							
D. PCOL	1-34	10-33*	11-28	18-53*	20-5*	20-17*	21-5*	22-11*	31-19*	32-23*	32-28*	33-18*
	33-43*	34-27*	35-8*	35-23*	39-50*	39-53*	39-60*	41-28*	43-24*	43-33*	45-25*	45-51*
	47-42*	48-42*	49-16*	51-17*	52-55*	53-16*	54-17*	55-17*	60-40*	61-21*	63-35*	65-17*
	65-18											
D. PFMT	1-35	3-32	40-16	63-22								
D. PS	1-30	3-30	4-17*	5-26*	7-35*	8-53*	8-56*	8-57*	8-61			
D. RO	1-27	3-22	5-27*	7-9*	7-28	8-49						
D. R1	1-27	3-23	7-40*	8-45								
D. R2	1-27	3-24	7-41*	8-44								
D. R3	1-27	3-25	7-42*	8-43								
D. R4	1-27	3-26	5-24*	5-44	7-43*	8-42						

D. R5	1-27	3-27	5-23*	5-45	7-44*	8-41						
D. R6	1-27	3-28	7-46*	8-39								
D. R7	1-27	3-29	4-16*	5-25*	5-37	7-8*	7-34*	7-70*	8-62	16-7*	17-26*	17-37
	26-73	27-14	27-61	27-66	29-62							
D. RLBS	1-25	3-34	13-23	14-23*	14-37	29-55	41-23	42-23	42-25	42-28		
D. SPSV	1-31	5-52	7-51*	35-9								
D. STAR	1-31	23-11										
D. SVCH	1-25	31-12	31-31*	31-41*								
D. V1FL	1-26	9-13*	9-22*	11-9	13-11	14-29	16-5	16-16	17-13	17-15*	18-57*	20-6
	20-20	21-6	22-15	36-13								
D. V2FL	1-26	9-15*	9-33*	14-13	15-5	16-27	16-40	19-15				
D. VAL1	1-26	9-14*	9-21*	13-10	14-23	14-31	15-19	16-7	16-18	17-21	19-6	19-9
	20-8	20-22	22-17	36-16								
D. VAL2	1-26	9-16*	9-32*	14-18	15-7	16-29	16-42	19-18				
DBGBRK	1-21	6-10#										
DBGENT	1-21	4-8#										
DBGINI	4-12	6-24	23-7#									
DBGOFF	8-11	44-13#										
DBGON	7-88	44-5#										
DBGRUN	4-21	5-46	7-39#									
DBGTRP	1-21	5-13#										
DECODE	11-24	27-67	45-12#									
DOPRT	5-31	5-39	9-5	12-9	16-8	16-19	17-65	20-32	21-14	21-19	22-22	22-27
	23-25	27-7	27-28	27-41	27-56	27-68	35-6	35-7	43-9#			
DP\$DAA	1-35	63-22										
DP\$LAA	1-35	40-16										
EM\$IIIV	1-89#	29-74										
EM\$IRB	1-90#	29-51										
EM\$IRC	1-86#	18-34										
EM\$IVC	1-87#	9-43										
EM\$IVL	1-88#	14-21	14-34	15-11								
EM\$NTL	1-91#	30-80										
ERRPRT	9-43	14-21	14-34	15-11	18-34	29-51	29-74	30-80	35-6#			
EXIT1	7-80	8-5#	16-10	16-20	17-66							
EXIT2	8-49#											
FLGBRK	15-25	28-7#										
GETCHR	9-26	9-37	18-13	18-24	18-39	18-54	29-19	29-53	29-68	30-17	30-30	30-71
	31-8#											
GETCMD	9-13#	10-15	10-35	11-35	13-35	14-57						
GETVAL	10-11	10-24	38-13#									
IB\$LEN	45-39	45-50	66-17#									
IB\$MSK	45-36	45-61	66-14#									
IB\$NAM	45-42	45-53	66-18#									
IB\$TYP	45-70	66-16#										
IB\$VAL	45-37	66-15#										
INSBAS	45-34	66-48#										
INSEND	66-166#											
INSLEN	17-57	17-70#										
INTADR	3-22#	29-76	39-30	39-47								
INTCHR	3-4#	3-17	29-70	39-51								
INTEND	3-35#	39-42										
LF	1-71#	1-92	2-20	18-17	18-27	18-42	18-55	43-31				
LSTADR	5-38	27-15	27-18	27-62	39-24	40-12#						
LSTTXT	5-34	5-36	9-9	10-14	10-28	10-34	12-19	12-25	13-17	13-20	13-31	14-28
	14-42	14-45	14-53	18-5	18-8	20-29	20-34	21-16	21-21	22-24	22-29	27-13
	27-16	27-19	27-34	27-46	27-49	27-57	35-20#	35-24	45-24			

PRTWRD	10-13	27-24	33-7#									
PSHCHR	9-29	18-14	18-47	29-24	30-24	30-50	30-72	31-41#				
PSW	1-32	5-53*	23-21*									
PUTCHR	1-32	43-23	43-30	43-32								
R50CHR	1-98#	18-29	18-32	18-35	34-18							
RELADR	41-18	42-14#	63-24									
SETLOC	10-6	10-21	11-11	36-13#								
SHOADR	13-30	14-52	40-21	41-10#	63-33							
SHOBRK	7-92	27-6#										
SHOLOC	12-18	12-24	20-28	20-33	21-15	21-20	22-23	22-28	39-11#			
SPACE	1-75#	65-16										
STRVAL	18-49	20-9	20-23	22-18	37-11#							
TAB	1-74#	2-23										
TM#GRT	1-92#	23-25										
TRCTRP	1-80#	7-25	7-27	7-58	8-23	8-53	8-56					
TSDBUG	1-6#											
TYP1	66-51	66-52	66-53	66-54	66-55	66-56	66-57	66-59	66-62	66-102	66-103	66-104
	66-105	66-137	66-138	66-139	66-140	66-141	66-165	66-179#				
TYP10	66-106	66-188#										
TYP11	66-115	66-116	66-189#									
TYP12	66-145	66-146	66-147	66-148	66-190#							
TYP13	66-149	66-150	66-151	66-152	66-153	66-154	66-156	66-162	66-191#			
TYP14	66-160	66-161	66-192#									
TYP15	66-155	66-193#										
TYP16	66-157	66-158	66-159	66-194#								
TYP2	66-58	66-65	66-73	66-75	66-76	66-77	66-78	66-79	66-80	66-81	66-82	66-83
	66-84	66-85	66-86	66-88	66-89	66-90	66-117	66-118	66-119	66-120	66-121	66-122
	66-123	66-124	66-125	66-126	66-127	66-128	66-129	66-130	66-142	66-143	66-144	66-163
	66-164	66-180#										
TYP3	66-60	66-181#										
TYP4	66-61	66-182#										
TYP5	66-63	66-64	66-183#									
TYP6	66-66	66-67	66-68	66-69	66-70	66-71	66-72	66-107	66-108	66-109	66-110	66-111
	66-112	66-113	66-114	66-184#								
TYP7	66-74	66-101	66-185#									
TYP8	66-87	66-97	66-98	66-99	66-100	66-186#						
TYP9	66-91	66-92	66-93	66-94	66-95	66-96	66-131	66-132	66-133	66-134	66-135	66-136
	66-187#											
TYPVEC	45-71	66-178#	66-179	66-180	66-181	66-182	66-183	66-184	66-185	66-186	66-187	66-188
	66-189	66-190	66-191	66-192	66-193	66-194						
UMODE	1-30	8-57										
UPMODE	1-30	1-32	5-53	8-57	23-21							

...	CM5	10-32	20-27	22-5	32-22	32-27	33-17	33-42	34-26	35-22	39-49	39-52	39-59
		41-27	45-55	47-29	47-32	47-35	47-38	48-41	49-15	50-16	51-16	52-54	53-15
		54-16	55-16	57-12	57-15	57-21	57-27	57-35	57-43	57-44	57-63	58-6	58-17
		58-18	58-36	60-18	60-19	60-26	60-27	60-32	60-36	61-12	61-13	61-17	63-31
		63-34	64-25	65-16									
	TTYIN	1-17#	31-18										
	TTYOU	1-17#	10-32	20-27	22-5	32-22	32-27	33-17	33-42	34-26	35-22	39-49	39-52
		39-59	41-27	45-55	47-29	47-32	47-35	47-38	48-41	49-15	50-16	51-16	52-54
		53-15	54-16	55-16	57-12	57-15	57-21	57-27	57-35	57-43	57-44	57-63	58-6
		58-17	58-18	58-36	60-18	60-19	60-26	60-27	60-32	60-36	61-12	61-13	61-17
		63-31	63-34	64-25	65-16								
ERR	1-45#	9-43	14-21	14-34	15-11	18-34	29-51	29-74	30-80				
INSTR	66-20#	66-51	66-52	66-53	66-54	66-55	66-56	66-57	66-58	66-59	66-60	66-61	
	66-62	66-63	66-64	66-65	66-66	66-67	66-68	66-69	66-70	66-71	66-72	66-73	
	66-74	66-75	66-76	66-77	66-78	66-79	66-80	66-81	66-82	66-83	66-84	66-85	
	66-86	66-87	66-88	66-89	66-90	66-91	66-92	66-93	66-94	66-95	66-96	66-97	
	66-98	66-99	66-100	66-101	66-102	66-103	66-104	66-105	66-106	66-107	66-108	66-109	
	66-110	66-111	66-112	66-113	66-114	66-115	66-116	66-117	66-118	66-119	66-120	66-121	
	66-122	66-123	66-124	66-125	66-126	66-127	66-128	66-129	66-130	66-131	66-132	66-133	
	66-134	66-135	66-136	66-137	66-138	66-139	66-140	66-141	66-142	66-143	66-144	66-145	
	66-146	66-147	66-148	66-149	66-150	66-151	66-152	66-153	66-154	66-155	66-156	66-157	
	66-158	66-159	66-160	66-161	66-162	66-163	66-164	66-165					
DCALL	1-40#	43-23	43-30	43-32									
PRINT	1-57#	5-34	5-36	9-9	10-14	10-28	10-34	12-19	12-25	13-17	13-20	13-31	
	14-28	14-42	14-45	14-53	18-5	18-8	20-29	20-34	21-16	21-21	22-24	22-29	
	27-13	27-16	27-19	27-34	27-46	27-49	27-57	45-24					
TPRINT	1-50#	5-31	5-39	9-5	12-9	16-8	16-19	17-65	20-32	21-14	21-19	22-22	
	22-27	23-25	27-7	27-28	27-41	27-56	27-68	35-6	35-7				
TYPDEF	66-171#	66-179	66-180	66-181	66-182	66-183	66-184	66-185	66-186	66-187	66-188	66-189	
	66-190	66-191	66-192	66-193	66-194								