

Table of contents

4-	1	DBGENT -- Enter debugger to start program
5-	1	DBCTRP -- 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	"L" -- 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
17-	1	"TAB" -- Single step the program
18-	1	"X" -- Display RAD50 value
19-	1	"N" -- Monitor data cell
20-	1	"CR" -- Close current location
20-	13	"LF" -- Close location and advance to next
21-	1	"J" -- 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	PRT OCT -- 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
60-	1	ODCREG -- Display register name

Table of contents

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  CBL
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*$STP, D*$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, $DBGBK, D*$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         ;
37         ; Macro definitions
38         ;
39         .MACRO  OCALL  ENTADD
40         CALL  OVRHC
41         .WORD  ENTADD
42         .ENDM  OCALL
43         ;
44         .MACRO  ERR  ERRMSG
45         MOV  ERRMSG, R1
46         JMP  ERRPRT
47         .ENDM  ERR
48         ;
49         .MACRO  TPRINT  STRING
50         .IF  NB STRING
51         MOV  STRING, R0
52         .ENDC
53         CALL  DOPRT
54         .ENDM  TPRINT
55         ;
56         .MACRO  PRINT  STRING
57         JSR  R2, LSTTXT

```

```

58          .ASCIZ  "'STRING'"
59          .EVEN
60          .ENDM  PRINT
61          ;
62          ; EMT argument blocks to control terminal debug mode
63          ;
64 000002    001    111    ONEMT:  .BYTE  1,111          ;Turn on debug mode
65 000004    000    111    OFFEMT: .BYTE  0,111          ;Turn off debug mode
66          ;
67          ; Ascii character values
68          ;
69          000015    CR      =      15          ;Carriage-return
70          000012    LF      =      12          ;Line-feed
71          000007    BELL    =      07          ;Bell
72          000010    BKSPAC  =      10          ;Backspace
73          000011    TAB     =      11          ;Tab
74          000040    SPACE   =      40          ;Space
75          000054    COMMA   =      54          ;Comma
76          ;
77          ; Assembly parameters
78          ;
79          000020    TRCTRP  =      20          ;Trace trap flag in PSW
80          000010    COLOPN  =      8          ;Column number where operand is to go
81          ;
82          ; Error messages
83          ;
84          .NLIST  BEX
85 000006    111    156    166    EM$IRC: .ASCIZ  /Invalid RAD50 character/
86 000036    111    156    166    EM$IVC: .ASCIZ  /Invalid command/
87 000056    111    156    166    EM$IVL: .ASCIZ  /Invalid value/
88 000074    111    156    166    EM$IIV: .ASCIZ  /Invalid register/
89 000115    111    156    166    EM$IRB: .ASCIZ  /Invalid relocation register/
90 000151    116    165    155    EM$NTL: .ASCIZ  /Number too long/
91 000171    015    012    124    TM$GRT: .ASCIZ  <CR><LF>/TSX-Plus debugger/
92 000215    000
93 000216    015    200
94          ;
95          ; RAD50 character table
96          ;
97 000220    040    101    102    R50CHR: .ASCII  / ABCDEFGHIJKLMNOPQRSTUVWXYZ$. *0123456789/
98          .EVEN
99          .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 'M' ; M
15 . BYTE 'P' ; P
16 . BYTE 'R' ; R
17 . BYTE 'S' ; S
18 . BYTE 'X' ; X
19 . BYTE CR ; Carriage-return
20 . BYTE LF ; Line-feed
21 . BYTE '^' ; Up arrow
22 . BYTE BKSPAC ; Backspace
23 . BYTE TAB ; Tab
24 NUMCMD = .-CMDTBL
25 . EVEN
26 ;
27 ; Parallel table of command processing routines
28 ;
29 CMDVEC: . WORD CMDSLH ; /
30 . WORD CMDBKS ; \
31 . WORD CMDAT ; @
32 . WORD CMDUS ; -
33 . WORD CMDEXP ; !
34 . WORD CMDDCD ; [
35 . WORD CMDRSB ; ]
36 . WORD CMDB ; B
37 . WORD CMDG ; G
38 . WORD CMDM ; M
39 . WORD CMDP ; P
40 . WORD CMDR ; R
41 . WORD CMDS ; S
42 . WORD CMDX ; X
43 . WORD CMDCR ; Carriage-return
44 . WORD CMDLF ; Line-feed
45 . WORD CMDUP ; Up arrow
46 . WORD CMDDBSP ; Backspace
47 . WORD CMDTAB ; Tab

```

```

1
2 ; -----
3 ; Table of internal register names
4 INTCHR: .BYTE '0 ; $0
5 .BYTE '1 ; $1
6 .BYTE '2 ; $2
7 .BYTE '3 ; $3
8 .BYTE '4 ; $4
9 .BYTE '5 ; $5
10 .BYTE '6 ; $6 (SP)
11 .BYTE '7 ; $7 (PC)
12 .BYTE 'S ; $S (PS)
13 .BYTE 'M ; $M (Mask word)
14 .BYTE 'F ; $F (Format register)
15 .BYTE 'B ; $B (Breakpoint locations)
16 .BYTE 'R ; $R (Relocation base addresses)
17 NUMINT = .-INTCHR
18 .EVEN
19 ;
20 ; Parallel table of addresses of internal register cells
21 ;
22 INTADR: .WORD D. R0 ; $0
23 .WORD D. R1 ; $1
24 .WORD D. R2 ; $2
25 .WORD D. R3 ; $3
26 .WORD D. R4 ; $4
27 .WORD D. R5 ; $5
28 .WORD D. R6 ; $6 (SP)
29 .WORD D. R7 ; $7 (PC)
30 .WORD D. PS ; $S (PS)
31 .WORD D. MASK ; $M (Mask word)
32 .WORD D. PFMT ; $F (Format register)
33 .WORD D. BKAD ; $B (Breakpoint locations)
34 .WORD D. RLBS ; $R (Relocation base addresses)
35 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 000432   DBGENT:
9          ;
10         ;  Initialize the debugger
11         ;
12 000432   004737   004170'   CALL    DBGINI           ;Initialize the debugger
13         ;
14         ;  Set starting address and initial PSW
15         ;
16 000436   012637   0000000   MOV     (SP)+,D,R7       ;Set starting address
17 000442   012637   0000000   MOV     (SP)+,D,PS      ;Set starting PSW
18         ;
19         ;  Enter debugger
20         ;
21 000446   000137   001052'   JMP     DBGRUN          ;Enter debugger
```

DBGTRP -- Entry from trap

```

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 000452  DBGTRP:
14         ;
15         ;  See if the trap occurred in the debugger or in the user's program
16         ;
17 000452  005704      TST      R4          ;Trap in debugger or user's program?
18 000454  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 000456  012637  0000000  MOV      (SP)+, D, R5
24 000462  012637  0000000  MOV      (SP)+, D, R4
25 000466  012637  0000000  MOV      (SP)+, D, R7      ;Trap PC
26 000472  012637  0000000  MOV      (SP)+, D, PS     ;Trap PSW
27 000476  010037  0000000  MOV      R0, D, R0
28         ;
29         ;  Print trap entry message
30         ;
31 000502          TPRINT  #CRLF          ;Go to a new line
32 000512  020527  0000001  CMP      R5, #1          ;Trap to 4 or 10?
33 000516  001017          BNE      1$              ;Br if trap to 10
34 000520          PRINT   <Trap to 4 at location >
35 000554  000416          BR      2$
36 000556          1$: PRINT   <Trap to 10 at location >
37 000612  013700  0000000  2$: MOV   D, R7, R0      ;Get address where trap occurred
38 000616  004737  007146'  CALL   LSTADR          ;Display the address
39 000622          4$: TPRINT #CRLF          ;Go to new line
40         ;
41         ;  Enter the debugger
42         ;
43 000632  042737  0000000  0000000  BIC     #<D$DBRK!D$IBRK!D$SBRK>, D, FLAG ;Say no breakpoint occurred
44 000640  013704  0000000  MOV     D, R4, R4      ;Restore R4 and R5
45 000644  013705  0000000  MOV     D, R5, R5
46 000650  000137  001052'  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 000654  013706  0000000  10$: MOV  D, SPSV, SP   ;Restore the stack pointer
53 000660  052737  0000000  0000000  BIS   #UPMODE, @#PSW  ;Make sure previous mode = user
54 000666  005037  0000000  CLR   D, LOC          ;No currently open cell
55 000672  000137  001506'  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         000676  DBGBRK:
11         ;
12         ;  Reset job control flag that was used to force the breakpoint
13         ;
14         000676  010146          MOV     R1,-(SP)
15         000700  113701  00000000  MOVB   CORUSR,R1      ;Get job index number
16         000704  042761  00000000 00000000  BIC   ##DBGBK,LSW9(R1);Clear flag saying to force debugger break
17         000712  052761  00000000 00000000  BIS   ##DEBUG,LSW9(R1);Say program is now running with debugger
18         000720  012601          MOV     (SP)+,R1
19         ;
20         ;  If we have not initialized the debugger, do the debugger startup
21         ;
22         000722  032737  00000000 00000000  BIT   #D$INIT,D.FLAG ;Has initialization been done?
23         000730  001002          BNE   1$             ;Br if yes
24         000732  004737  004170'   CALL  DBGINI         ;Initialize the debugger
25         ;
26         ;  Enter debugger as if we hit a breakpoint
27         ;
28         000736  052737  00000000 00000000 1$:  BIS   #D$FBRK,D.FLAG ;Set flag saying we had a forced break
29         000744  000137  000750'   JMP   BRKENT        ;Now enter debugger as if we had a breakpoint

```

BRKENT -- Enter debugger from 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 000750 011637 000000G BRKENT: MOV      (SP),D.R7      ;Save breakpoint PC
9 000754 010037 000000G      MOV      R0,D.R0      ;Save user's R0
10         ;
11         ; Determine if we should enter the debugger
12         ;
13 000760 004737 004420'      CALL     BRKCHK      ;See if breakpoint causes program stop
14 000764 032737 000000C 000000G BIT      #<D$DBRK!D$!BRK!D$SBRK!D$FBRK>,D.FLAG ;Breakpoint?
15 000772 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 000774 042737 000000G 000000G BIC      #D$IPND,D.FLAG ;Say pending instruction has been executed
22 001002 032737 000000C 000000G BIT      #<D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Are we single stepping
23 001010 001006          BNE      3$          ;Br if yes
24 001012 004737 004340'      CALL     BRKSET      ;Set breakpoints
25 001016 042766 000020 000002 BIC      #TRCTRP,2(SP) ;Reset trace trap flag in PSW
26 001024 000403          BR       2$          ;
27 001026 052766 000020 000002 3$: BIS      #TRCTRP,2(SP) ;Set trap flag in PSW on stack
28 001034 013700 000000G 2$: MOV      D.R0,R0      ;Recover R0
29 001040 000006          RTT          ;Return from breakpoint
30         ;
31         ; This breakpoint will enter the debugger.
32         ; Save information and enter the debugger
33         ;
34 001042 012637 000000G 5$: MOV      (SP)+,D.R7      ;Save trap PC
35 001046 012637 000000G      MOV      (SP)+,D.PS      ;Save trap PS
36         ;
37         ; Save user's registers
38         ;
39 001052 052737 000000G 000000G DBGRUN: BIS      #D$RUN,D.FLAG ;Set flag saying debugger is running
40 001060 010137 000000G      MOV      R1,D.R1
41 001064 010237 000000G      MOV      R2,D.R2
42 001070 010337 000000G      MOV      R3,D.R3
43 001074 010437 000000G      MOV      R4,D.R4
44 001100 010537 000000G      MOV      R5,D.R5
45 001104 106506          MFPD     SP
46 001106 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 001112 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 001116 106537 000036          MFPD     @#36          ;Get user's TRAP PSW

```

BRKENT -- Enter debugger from breakpoint

```

58 001122 042716 000020          BIC    #TRCTR, (SP)    ; Clear the trace-trap flag
59 001126 106637 000036          MTPD   @#36           ; Store new TRAP PSW
60                               ;
61                               ; If breakpoints are set in program, restore the real instructions
62                               ; that belong in the breakpoint locations.
63                               ;
64 001132 042737 000000 000000    BIC    #D$IPND, D.FLAG ; Clear instruction-pending flag
65 001140 032737 000000 000000    BIT    #<D$DMON!D$SSTP!D$TSTP>, D.FLAG ; Are we single stepping?
66 001146 001014                   BNE    2$             ; Br if yes
67 001150 004737 004244'          CALL   BRKRST         ; Replace breakpoint instructions
68 001154 032737 000000 000000    BIT    #D$IBRK, D.FLAG ; Did an instruction breakpoint occur?
69 001162 001406                   BEQ    2$             ; Br if not
70 001164 162737 000002 000000    SUB    #2, D.R7       ; Backup PC to point to BPT instruction
71 001172 052737 000000 000000    BIS    #D$IPND, D.FLAG ; Remember instruction must be executed
72                               ;
73                               ; See if a proceed repeat count is in effect
74                               ;
75 001200 005737 000000          2$:   TST    D.PCNT     ; Is a proceed repeat count in effect?
76 001204 001407                   BEQ    3$             ; Br if not
77 001206 032737 000000 000000    BIT    #D$FBRK, D.FLAG ; Is this a forced breakpoint?
78 001214 001003                   BNE    3$             ; Br if yes -- always stop
79 001216 005337 000000          DEC    D.PCNT     ; Should we skip this breakpoint?
80 001222 001011                   BNE    EXIT1         ; Br if yes
81                               ;
82                               ; Clear temporary single-step flag
83                               ;
84 001224 042737 000000 000000    3$:   BIC    #D$TSTP, D.FLAG ; Clear temporary single-step flag
85                               ;
86                               ; Put terminal control in debug mode
87                               ;
88 001232 004737 007464'          CALL   DBGON         ; Put terminal in debug mode
89                               ;
90                               ; Display information about the breakpoint
91                               ;
92 001236 004737 004632'          CALL   SHOBRK        ; Display breakpoint information
93 001242 000137 001506'          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 001246  042737  0000000 0000000 EXIT1:  BIC      #D#FBRK,D.FLAG ;Turn off forced-breakpoint flag
6 001254  032737  0000000 0000000        BIT      #<D#SSTP!D#TSTP!D#DMON>,D.FLAG ;Are we single stepping?
7 001262  001403                BEQ      3$          ;Br if not
8 001264  042737  0000000 0000000        BIC      #D#IPND,D.FLAG ;Turn of instruction-pending if single step
9 001272  005037  0000000                3$:   CLR      D.LOC      ;Say no location is open
10 001276  042737  0000000 0000000        BIC      #D#RUN,D.FLAG  ;Say debugger is no longer running
11 001304  004737  007474'        CALL    DBGOFF     ;Turn off terminal debug mode
12 001310  032737  0000000 0000000        BIT      #<D#IPND!D#DMON!D#SSTP!D#TSTP>,D.FLAG ;Should we single step?
13 001316  001003                BNE      4$          ;Br if yes
14 001320  004737  004340'        CALL    BRKSET     ;Set breakpoints
15 001324  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 001326  106537  000036        4$:   MFPD    @#36          ;Get user's TRAP PSW
23 001332  052716  000020                BIS      #TRCTRP,(SP) ;Set the trace-trap flag in the PSW
24 001336  106637  000036                MTPD    @#36          ;Store new TRAP PSW
25          ;
26          ; If we are doing data monitoring, save current value of monitored cell
27          ;
28 001342  032737  0000000 0000000 1$:   BIT      #D#DMON,D.FLAG ;Are we doing data cell monitoring?
29 001350  001411                BEQ      2$          ;Br if not
30 001352  013705  0000000                MOV     D.DADR,R5    ;Get address of cell being monitored
31 001356  106515                MFPD    (R5)         ;Get current value from cell
32 001360  013704  0000000                MOV     D.MASK,R4    ;Get current data mask
33 001364  005104                COM     R4           ;Complement
34 001366  040405                BIC     R4,R5        ;Clear all but bits of interest
35 001370  012637  0000000                MOV     (SP)+,D.DOLD ;Save old value
36          ;
37          ; Restore user's registers
38          ;
39 001374  013746  0000000        2$:   MOV     D.R6,-(SP)
40 001400  106606                MTPD    SP
41 001402  013705  0000000                MOV     D.R5,R5
42 001406  013704  0000000                MOV     D.R4,R4
43 001412  013703  0000000                MOV     D.R3,R3
44 001416  013702  0000000                MOV     D.R2,R2
45 001422  013701  0000000                MOV     D.R1,R1
46          ;
47          ; Exit point used if we did not enter debugger
48          ;
49 001426  013700  0000000        EXIT2: MOV     D.R0,R0
50          ;
51          ; See if we need to set trap flag in PS
52          ;
53 001432  042737  000020 0000000        BIC     #TRCTRP,D.PS ;Reset trace trap flag in PS
54 001440  032737  0000000 0000000        BIT     #<D#IPND!D#DMON!D#SSTP!D#TSTP>,D.FLAG ;Do we want single step?
55 001446  001403                BEQ     1$          ;Br if not
56 001450  052737  000020 0000000        BIS     #TRCTRP,D.PS ;Set Trap flag in user's PS
57 001456  052737  0000000 0000000 1$:   BIS     #UMODE!UPMODE,D.PS ;Make sure user-mode is set in PS

```

```
58 ;  
59 ; Use RTT instruction to reenter user's program  
60 ;  
61 001464 013746 0000000 MOV D.PS, -(SP) ;Set PS for user  
62 001470 013746 0000000 MOV D.R7, -(SP) ;Set PC for user  
63 001474 000006 RTT ;Enter user's program
```

NEWCMD -- Get new command

```

1          .SBTTL  NEWCMD -- Get new command
2          ;-----
3          ; Print carriage-return, line-feed and get new command
4          ;
5 001476   NEWLIN: TPRINT  #CRLF
6          ;
7          ; Get new command
8          ;
9 001506   NEWCMD: PRINT  <DBG:>
10         ;
11         ; Assume neither value1 nor value2 is specified
12         ;
13 001520   105037  0000000  GETCMD: CLR  D, V1FL      ; Value1 not specified
14 001524   005037  0000000          CLR  D, VAL1
15 001530   105037  0000000          CLR  D, V2FL      ; Value2 not specified
16 001534   005037  0000000          CLR  D, VAL2
17         ;
18         ; See if Value1 is specified
19         ;
20 001540   004737  005252'   CALL  ACRVAL      ; Try to accrue value1
21 001544   010037  0000000   MOV   RO, D, VAL1   ; Save value for value1
22 001550   110137  0000000   MOVB  R1, D, V1FL   ; Remember if value1 specified
23         ;
24         ; See if value2 is specified
25         ;
26 001554   004737  005720'   CALL  GETCHR      ; Get delimiter
27 001560   120027  000073   CMPB  RO, #'      ; is there a value2?
28 001564   001403          BEQ   1$           ; Br if possibly yes
29 001566   004737  005764'   CALL  PSHCHR      ; Save command character
30 001572   000404          BR    2$           ;
31 001574   004737  005252'   1$: CALL  ACRVAL      ; Get value2
32 001600   010037  0000000   MOV   RO, D, VAL2   ; Save value2
33 001604   110137  0000000   MOVB  R1, D, V2FL   ; Save info about value2
34         ;
35         ; Look up command character
36         ;
37 001610   004737  005720'   2$: CALL  GETCHR      ; Get command character
38 001614   012701  000023   MOV   #NUMCMD, R1   ; Get # valid command characters
39 001620   120061  000270'   3$: CMPB  RO, CMDTBL(R1) ; Search for character in table
40 001624   001406          BEQ   4$           ; Br if found command character
41 001626   005301          DEC  R1           ; More command chars to check?
42 001630   002373          BGE  3$           ; Loop if yes
43 001632          ERR   #EM#IVC ; Invalid command character
44         ;
45         ; Found command character.
46         ; Jump off to processing routine.
47         ;
48 001642   006301          4$: ASL  R1           ; Convert command index into word table index
49 001644   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 001650 004737 006370'  CMDSLH: CALL  SETLOC      ;Set value1 as current location
7 001654 032737 000001 000000G CMDSL1: BIT    #1,D.LOC    ;Is location odd?
8 001662 001021          BNE    CMDBKS      ;If yes then treat "/" like "\"
9 001664 105037 000000G  CLR    D.BYTM     ;Say we are in word mode
10 001670 012737 000002 000000G  MOV    #2,D.ILEN   ;Say instruction length = 2 bytes
11 001676 004737 006614'  CALL  GETVAL      ;Get contents of location
12 001702 010037 000000G  MOV    R0,D.LVAL   ;Save last value displayed
13 001706 004737 006100'  CALL  PRTWRD     ;Print the value
14 001712          PRINT  < >      ;Print two spaces
15 001722 000137 001520'  JMP    GETCMD     ;Go get next command
16
17          .SBTTL  "\"  -- Examine byte
18          ;-----
19          ; "address\" -- Examine contents of a byte
20          ;
21 001726 004737 006370'  CMDBKS: CALL  SETLOC      ;Set value1 as current location
22 001732 112737 000001 000000G CMDBK1: MOVB  #1,D.BYTM   ;Say we are operating in byte mode
23 001740 112737 000001 000000G  MOVB  #1,D.ILEN   ;Say instruction length = 1 byte
24 001746 004737 006614'  CALL  GETVAL      ;Get contents of location
25 001752 010037 000000G  MOV    R0,D.LVAL   ;Save last value displayed
26 001756 010002          MOV    R0,R2        ;Save value
27 001760 004737 006154'  CALL  PRTBYT     ;Print value
28 001764          PRINT  < = >      ;Put in equal sign
29 001774 120227 000040  CMPB  R2,#40      ;Is this a printing character?
30 002000 103002          BHS    1$         ;Br if yes
31 002002 112702 000056  MOVB  #',R2      ;Use period for non-printing chars
32 002006          1$: .TTYOUT R2    ;Print character
33 002014 005237 000000G  INC    D.PCOL     ;Advance print column
34 002020          PRINT  < >      ;Print 2 spaces
35 002030 000137 001520'  JMP    GETCMD     ;Go get next command

```

"I" -- Decode value as an instruction

```

1          .SBTTL  "I"  -- Decode value as an instruction
2          ;-----
3          ;  Open a cell and display its contents as an instruction.
4          ;
5 002034  CMDDCD:
6          ;
7          ;  See if an address was specified
8          ;
9 002034  105737  0000000          TSTB   D.V1FL          ;Was an address specified?
10 002040  001402          BEQ    1$              ;Br if not
11 002042  004737  006370'      CALL   SETLOC         ;Set address of currently open cell
12 002046  013701  0000000  1$:  MOV    D.LOC,R1      ;Is a cell currently open?
13 002052  001002          BNE    2$              ;Br if yes
14 002054  000137  001476'      JMP    NEWLIN         ;Nothing to decode
15 002060  032701  0000001  2$:  BIT    #1,R1        ;Is the address odd?
16 002064  001402          BEQ    3$              ;Br if not
17 002066  000137  001726'      JMP    CMDBKS         ;Treat like "\" if odd
18 002072  105737  0000000  3$:  TSTB   D.LOCM         ;Is cell internal or external?
19 002076  002002          BGE    4$              ;Br if external
20 002100  000137  001654'      JMP    CMDSL1        ;Treat I like / for internal cell
21          ;
22          ;  Decode the instruction
23          ;
24 002104  004737  007510'      4$:  CALL   DECODE         ;Decode the instruction
25          ;
26          ;  Tab over some
27          ;
28 002110  013700  0000000          MOV    D.PCOL,RO     ;Get current print column
29 002114  062700  000010          ADD    #8.,RO        ;Tab over some
30 002120  042700  0000007          BIC    #7,RO
31 002124  004737  011734'      CALL   ODCTAB        ;Tab over
32          ;
33          ;  Finished
34          ;
35 002130  000137  001520'      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 002134  013702  0000000  CMDAT:  MOV      D.LVAL,R2      ;Get last displayed value
7 002140  010237  0000000  CMDAT1: MOV     R2,D.LOC      ;Set as new location to display
8 002144  112737  000001  0000000  MOVB    #1,D.LO这里CM      ;Say new address is in user's program
9 002152          TPRINT  #CRLF      ;Go to a new line
10         ;
11         ; See if we should do the display in byte or word mode
12         ;
13 002162  032702  000001          BIT      #1,R2      ;Should we display in byte or word mode?
14 002166  001010          BNE     1$          ;Br if need to go to byte mode
15         ;
16         ; Display new location in word mode
17         ;
18 002170  004737  006750'          CALL    SHLOLOC      ;Display the current address
19 002174          PRINT  < / >      ;Display "/"
20 002204  000137  001654'          JMP     CMDSL1      ;Simulate the "/" command
21         ;
22         ; Display in byte mode
23         ;
24 002210  004737  006750' 1$:  CALL    SHLOLOC      ;Display the current address
25 002214          PRINT  < \ >      ;Display "\"
26 002224  000137  001732'          JMP     CMDBK1      ;Simulate the "\" command
27         ;
28         .SBTTL  "_"  -- Open cell indirect, relative to PC
29         ;-----
30         ; "_" -- Open indirect relative to PC.
31         ;
32 002230  013702  0000000  CMDUS:  MOV      D.LVAL,R2      ;Get last displayed value
33 002234  063702  0000000          ADD     D.LOC,R2      ;Add PC of instruction
34 002240  062702  000002          ADD     #2,R2        ;Plus 2
35 002244  000137  002140'          JMP     CMDAT1      ;Now treat like "@" command

```

!" -- Convert address to relocation offset

```

1          .SBTTL  !" -- Convert address to relocation offset
2          ;-----
3          ;  !" -- Convert current address to relocation offset
4          ;
5 002250   CMDEXP:
6          ;
7          ;  If no relocation register number was provided, find closest relocation
8          ;  base.
9          ;
10 002250  013701  0000000  MOV     D.VAL1,R1      ;Get relocation register #
11 002254  105737  0000000  TSTB   D.V1FL        ;Was a relocation register specified?
12 002260  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 002262                PRINT  < = >      ;Print "="
18 002272  010100  MOV     R1,RO         ;Get the relocation register number
19 002274  004737  005772'  CALL   PRTOCT        ;Print the relocation reg #
20 002300                PRINT  < , >      ;Print ", "
21 002306  006301  ASL     R1            ;Convert relocation # to word table index
22 002310  013700  0000000  MOV     D.LOC,RO     ;Get original value
23 002314  166100  0000000  SUB     D.RLBS(R1),RO ;Subtract relocation base
24 002320  004737  005772'  CALL   PRTOCT        ;Print the offset
25 002324  000404  BR      B$
26          ;
27          ;  No relocation register number was specified
28          ;
29 002326  013700  0000000  2$:    MOV     D.LOC,RO ;Get address
30 002332  004737  007172'  CALL   SHADR        ;Display it with best relocation base
31 002336                B$:    PRINT  < >      ;Print 2 spaces
32          ;
33          ;  Finished
34          ;
35 002346  000137  001520'  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 002352   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 002352   105737 0000000   TSTB   D.V2FL      ;Was argument 2 specified?
14 002356   001417          BEQ     2$          ;Br if not
15         ;
16         ; Set base address for relocation area
17         ;
18 002360   013701 0000000   MOV    D.VAL2,R1      ;Get register number
19 002364   020127 0000007   CMP   R1,#7         ;Should not exceed 7
20 002370   101404          BLOS  1$            ;Br if ok
21 002372          ERR    #EM#IVL
22 002402   006301          1$:   ASL   R1          ;Convert to word table index
23 002404   013761 0000000 0000000   MOV   D.VAL1,D.RLBS(R1);Set address for relocation base
24 002412   000137 001476'   JMP   NEWLIN        ;Go get next command
25         ;
26         ; Calculate offset relative to relocation base
27         ;
28 002416          2$:   PRINT  < = >      ;Print "="
29 002426   105737 0000000   TSTB  D.V1FL       ;Was a relocation reg # provided?
30 002432   001436          BEQ   4$            ;Br if not
31 002434   013701 0000000   MOV   D.VAL1,R1     ;Get register number
32 002440   020127 0000007   CMP   R1,#7         ;Should not exceed 7
33 002444   101404          BLOS  3$            ;Br if ok
34 002446          ERR    #EM#IVL
35 002456   006301          3$:   ASL   R1          ;Convert to word table index
36 002460   013700 0000000   MOV   D.LVAL,R0     ;Get currently open value
37 002464   166100 0000000   SUB   D.RLBS(R1),R0 ;Calculate offset
38 002470   010002          MOV   R0,R2         ;Hold the offset value
39 002472   010100          MOV   R1,R0         ;Get offset region #
40 002474   006200          ASR   R0            ;Convert to ordinal number
41 002476   004737 005772'   CALL  PRTOCT        ;Print relocation #
42 002502          PRINT  < , >      ;Print comma
43 002510   010200          MOV   R2,R0         ;Get offset value
44 002512   004737 005772'   CALL  PRTOCT        ;Print it
45 002516          PRINT  < >        ;Print a space
46 002526   000410          BR   9$            ;
47         ;
48         ; No relocation register number was provided
49         ; Show address with best relocation
50         ;
51 002530   013700 0000000          4$:   MOV   D.LVAL,R0     ;Get value of interest
52 002534   004737 007172'          CALL  SHOADR        ;Display the address
53 002540          PRINT  < >        ;Print 2 spaces
54         ;
55         ; Finished
56         ;
57 002550   000137 001520'          9$:   JMP   GETCMD        ;Go get next command

```

"B" -- Set breakpoint

```

1
2
3
4
5 002554 105737 0000000 CMDB: TSTB D.V2FL ;Was a breakpoint number specified?
6 002560 001412 BEQ 1$ ;Br if not
7 002562 013702 0000000 MOV D.VAL2,R2 ;Get breakpoint number
8 002566 006302 ASL R2 ;Convert to word table index
9 002570 020227 000016 CMP R2,#14. ;Make sure it's not too big
10 002574 101416 BLOS 3$ ;Br if ok
11 002576 ERR #EM#IVL
12 002606 005002 1$: CLR R2 ;Search for a free breakpoint entry
13 002610 005762 0000000 4$: TST D.BKAD(R2) ;Is this breakpoint free?
14 002614 001406 BEQ 3$ ;Br if yes
15 002616 062702 000002 ADD #2,R2 ;Point to next breakpoint entry
16 002622 020227 000016 CMP R2,#14. ;Have we checked all breakpoint entries?
17 002626 101770 BLOS 4$ ;Br if more to check
18 002630 005002 CLR R2 ;If all busy, then use # 0
19 002632 013701 0000000 3$: MOV D.VAL1,R1 ;Get address of breakpoint
20 002636 004737 007504' CALL CHKADR ;Make sure its ok
21 002642 010162 0000000 MOV R1,D.BKAD(R2) ;Save address of breakpoint
22
23 ; Set D$CKBK flag in D.FLAGS if there are any active breakpoints
24
25 002646 004737 005206' CALL FLGBRK ;Set flag if any breakpoints are active
26 002652 000137 001476' 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 002656 105737 0000000 CMDG:  TSTB  D.V1FL      ;Was a starting address specified?
6 002662 003403          BLE  1$          ;Br if not
7 002664 013737 0000000 0000000 MOV  D.VAL1,D.R7    ;Set starting address
8 002672          1#:  TPRINT #CRLF      ;Go to new line
9 002702 005037 0000000 CLR  D.PCNT      ;Clear any proceed count
10 002706 000137 001246' JMP  EXIT1       ;Start running program
11
12          .SBTTL "P" -- Proceed from breakpoint
13          ;-----
14          ; "n;P" -- Proceed from breakpoint
15          ;
16 002712 105737 0000000 CMDP:  TSTB  D.V1FL      ;Was a proceed repeat count specified?
17 002716 003403          BLE  1$          ;Br if not
18 002720 013737 0000000 0000000 MOV  D.VAL1,D.PCNT  ;Set proceed repeat count
19 002726          1#:  TPRINT #CRLF      ;Go to new line
20 002736 000137 001246' 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 002742 042737 0000000 0000000 CMDS:  BIC  #D$SSTP,D.FLAG ;Assume he wants to reset single-step mode
27 002750 105737 0000000          TSTB  D.V2FL      ;Was a value specified for "n"?
28 002754 001406          BEQ  1$          ;Br if not
29 002756 005737 0000000          TST  D.VAL2      ;Is value 2 non-zero?
30 002762 001403          BEQ  1$          ;If zero, don't single step
31 002764 052737 0000000 0000000 BIS  #D$SSTP,D.FLAG ;Set single-step mode
32 002772 000137 001476' 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 002776  CMDTAB:
10         ;
11         ; See if a value was specified with TAB
12         ;
13 002776 105737 0000000  TSTB   D.V1FL      ;Was value 1 specified?
14 003002 001407         BEQ     2$          ;Br if not
15 003004 105037 0000000  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 003010 013700 0000000 1$:   MOV    D.VAL1,R0      ;Get specified value
22 003014 001407         BEQ     5$          ;0 ==> Skip over subprogram call
23         ;
24         ; A starting address was specified.
25         ;
26 003016 010037 0000000  MOV    R0,D.R7        ;Set starting address
27         ;
28         ; Single step the program.
29         ;
30 003022 052737 0000000 0000000 2$:   BIS    #D#TSTP,D.FLAG ;Say we are in temporary single step mode
31 003030 000137 002712'  JMP    CMDP          ;Now treat tab like ;P
32         ;
33         ; Argument value 1 is 0 (zero).
34         ; Step over a subprogram call.
35         ; Make sure the current instruction is a CALL.
36         ;
37 003034 013701 0000000 5$:   MOV    D.R7,R1        ;Get address of current instruction
38 003040 106521         MFPD   (R1)+          ;Get instruction from user's program
39 003042 012602         MOV    (SP)+,R2       ;Get the instruction from the stack
40 003044 010200         MOV    R2,R0          ;Copy the instruction word
41 003046 042700 000777  BIC    #777,R0        ;Clear register # and address field
42 003052 020027 004000  CMP    R0,#0004000    ;Is this a JSR instruction?
43 003056 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 003060 010200         MOV    R2,R0          ;Get the instruction
49 003062 042700 177770  BIC    #^C7,R0        ;Clear all but addressing register #
50 003066 020027 000007  CMP    R0,#7          ;Is it using register 7 (PC)?
51 003072 001003         BNE    3$            ;Br if not
52 003074 062701 000002  ADD    #2,R1          ;2 bytes for subroutine address if PC used
53 003100 000410         BR     4$
54 003102 010200 3$:   MOV    R2,R0          ;Get back the instruction
55 003104 042700 177707  BIC    #^C70,R0       ;Clear all but addressing mode
56 003110 072027 177775  ASH    #-3,R0         ;Right justify the addressing mode
57 003114 116000 003156' MOVB   INSLN(R0),R0    ;Get # bytes used by this addressing mode

```

"TAB"-- Single step the program

```

58 003120 060001          ADD    R0,R1          ;Get address of location where breakpoint goes
59 003122 010137 000000G 4#:    MOV    R1,D.CBRK      ;Set address of call break point
60 003126 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 003134 042737 000000G 000000G  BIC    #D#SSTP,D.FLAG ;Reset single step mode
65 003142                                     TPRINT #CRLF          ;Go to a new line
66 003152 000137 001246'          JMP    EXIT1          ;Enter user's program
67                                     ;
68                                     ; Table providing number of bytes used by each addressing mode
69                                     ;
70 003156          000      000      000  INSLEN: .BYTE 0,0,0,0,0,0,2,2
      003161          000      000      000
      003164          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 003166   CMDX:  PRINT  < = >          ;Print "="
6 003176   013700 0000000   MOV      D.LVAL,R0          ;Get last displayed value
7 003202   004737 006232'   CALL     PRTR50           ;Convert to ascii and print
8 003206   PRINT  < >          ;Print spaces
9 003216   012737 000002 0000000  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 003224   004737 005720'   CALL     GETCHR          ;Get 1st character
14 003230   004737 005764'   CALL     PSHCHR          ;Push the character
15 003234   120027 000015   CMPB    RO,#CR          ;No new value wanted?
16 003240   001454   BEQ      7$             ;Br if no new value specified
17 003242   120027 000012   CMPB    RO,#LF          ;Advance to next cell?
18 003246   001451   BEQ      7$             ;Br if yes
19
20          ; Now accept a new RAD50 value
21          ;
22 003250   005001   CLR      R1              ;Form new RAD50 value in R1
23 003252   012702 000003   MOV     #3,R2           ;Set count of # chars to accrue
24 003256   004737 005720'   1$: CALL     GETCHR          ;Get next character of value
25 003262   120027 000015   CMPB    RO,#CR          ;Reached end?
26 003266   001434   BEQ     2$             ;Br if yes
27 003270   120027 000012   CMPB    RO,#LF          ;LF is also end
28 003274   001431   BEQ     2$
29 003276   012703 000220'   MOV     #R50CHR,R3      ;Point to RAD50 character table
30 003302   120023   3$: CMPB    RO,(R3)+       ;Search for char in table
31 003304   001407   BEQ     4$             ;Br if found
32 003306   020327 000270'   CMP     R3,#R50CHR+50   ;Checked all of table?
33 003312   103773   BLO     3$             ;Loop if not
34 003314   ERR     #EM#IRC        ;Invalid RAD50 character
35 003324   162703 000221'   4$: SUB     #R50CHR+1,R3  ;Get RAD50 character value
36 003330   070127 000050   MUL     #50,R1          ;Multiply previous value by 50
37 003334   060301   ADD     R3,R1           ;Add new character value
38 003336   077231   SOB     R2,1$          ;Loop if more characters needed
39 003340   004737 005720'   6$: CALL     GETCHR          ;Ignore other characters up to CR
40 003344   120027 000015   CMPB    RO,#CR
41 003350   001403   BEQ     2$
42 003352   120027 000012   CMPB    RO,#LF
43 003356   001370   BNE     6$
44
45          ; We have gotten the RAD50 value. Store it in currently open cell
46          ;
47 003360   004737 005764'   2$: CALL     PSHCHR          ;Save the terminating character
48 003364   010100   MOV     R1,R0           ;Get value to R0 for STRVAL
49 003366   004737 006412'   CALL     STRVAL          ;Store value into currently open cell
50
51          ; If command ended with Line feed, open next cell
52          ;
53 003372   005037 0000000   7$: CLR     D.PCOL        ;Say print column = 0
54 003376   004737 005720'   CALL     GETCHR          ;Get the terminating character
55 003402   120027 000012   CMPB    RO,#LF          ;Open the next cell?
56 003406   001004   BNE     9$             ;Br if not
57 003410   105037 0000000   CLRB    D.V1FL         ;Tell LF not to store into current cell

```


"X" -- Display RAD50 value

```
58 003414 000137 003536'          JMP      CMDLF          ;Go open the next cell
59                                     ;
60                                     ; Finished
61                                     ;
62 003420 000137 001476'          9#:     JMP      NEWLIN
```

"M" -- Monitor data cell

```

1          .SBTTL  "M"  -- Monitor data cell
2          ;-----
3          ; "address;valueM" -- Monitor data cell
4          ;
5 003424  042737  0000000 0000000 CMDM:  BIC      #<D$DMON!D$DVAL>,D.FLAG ;Reset data-cell monitoring flags
6 003432  005737  0000000          TST      D.VAL1      ;Was an address specified?
7 003436  001420                   BEQ      9$          ;Br if not -- Disable monitoring
8 003440  052737  0000000 0000000          BIS      #D$DMON,D.FLAG ;Enable data monitoring
9 003446  013700  0000000          MOV      D.VAL1,RO    ;Get address of cell to monitor
10 003452  010037  0000000          MOV      RO,D.DADR   ;Set it as control address
11          ;
12          ; See if breakpoint is to be triggered whenever value in cell changes
13          ; or only when a specified value occurs.
14          ;
15 003456  105737  0000000          TSTB   D.V2FL      ;Was a value specified?
16 003462  003406                   BLE      9$          ;Br if not
17 003464  052737  0000000 0000000          BIS      #D$DVAL,D.FLAG ;Watch for specified value
18 003472  013737  0000000 0000000          MOV      D.VAL2,D.DTRC ;Save the specified value
19          ;
20          ; Finished
21          ;
22 003500  000137  001476'          9$:    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 003504 005037 0000000 CMDCR: CLR      D. PCOL      ; Say print column = 0
6 003510 105737 0000000      TSTB     D. V1FL     ; Was a new value specified for this cell?
7 003514 003404          BLE      9$         ; Br if not
8 003516 013700          MOV      D. VAL1, R0 ; Yes, get the value
9 003522 004737 006412'    CALL     STRVAL    ; Store value into current location
10 003526 005037 0000000 9$:  CLR      D. LOC      ; Say no location open now
11 003532 000137 001506'    JMP      NEWCMD   ; Go get next command
12
13          .SBTTL "LF" -- Close location and advance to next
14          ;-----
15          ; "line-feed" -- Close current cell and advance to next
16          ;
17 003536 005037 0000000 CMDLF: CLR      D. PCOL      ; Say print column = 0
18 003542 005737 0000000      TST      D. LOC      ; Is a location open now?
19 003546 001447          BEQ      5$         ; Br if not
20 003550 105737 0000000      TSTB     D. V1FL     ; Was a new value specified for this cell?
21 003554 003404          BLE      1$         ; Br if not
22 003556 013700          MOV      D. VAL1, R0 ; Get new value
23 003562 004737 006412'    CALL     STRVAL    ; Store into currently open cell
24 003566 105737 0000000 1$:  TSTB     D. BYTM     ; Are we in byte or word mode?
25 003572 001017          BNE      2$         ; Br if in byte mode
26 003574 062737 000002 0000000 ADD      #2, D. LOC ; Advance location counter
27 003602          .TTYOUT #CR      ; Go to new line
28 003612 004737 006750'    CALL     SHOLOC   ; Display address of new location
29 003616          PRINT    < / > ; Display "/"
30 003626 000137 001654'    JMP      CMDSL1   ; Simulate "/" command
31 003632 005237 0000000 2$:  INC      D. LOC      ; Advance to next byte
32 003636          TPRINT   #CRCHAR ; go to new line
33 003646 004737 006750'    CALL     SHOLOC   ; Display address of new location
34 003652          PRINT    < \ > ; Display "\"
35 003662 000137 001732'    JMP      CMDBK1   ; Simulate "\" command
36 003666 000137 001476'    5$:  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 003672 005037 0000000  CMDRSB: CLR      D.PCOL      ; Say print column = 0
6 003676 105737 0000000      TSTB     D.V1FL      ; Was an address specified?
7 003702 001402      BEQ      3$          ; Br if not
8 003704 000137 002034'    JMP      CMDDCD     ; Treat J like [
9 003710 005737 0000000  3$:  TST     D.LOC      ; Is a location open now?
10 003714 001440      BEQ      5$          ; Br if not
11 003716 105737 0000000      TSTB     D.BYTM     ; Are we in byte or word mode?
12 003722 001017      BNE      2$          ; Br if in byte mode
13 003724 063737 0000000 0000000  ADD     D.ILEN, D.LOC ; Advance location counter
14 003732      TPRINT  #CRLF     ; Go to new line
15 003742 004737 006750'    CALL    SHOLOC     ; Display address of new location
16 003746      PRINT   < [ >    ; Display "["
17 003756 000137 002034'    JMP     CMDDCD     ; Simulate "[" command
18 003762 005237 0000000  2$:  INC     D.LOC      ; Advance to next byte
19 003766      TPRINT  #CRCHAR   ; go to new line
20 003776 004737 006750'    CALL    SHOLOC     ; Display address of new location
21 004002      PRINT   < \ >    ; Display "\"
22 004012 000137 001726'    JMP     CMDBKS     ; Simulate "\" command
23 004016 000137 001476'    5$:  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 004022  CMDBSP: .TTYOUT #136          ;Print "^"
6 004032  000400          BR          CMDUP          ;Now treat backspace like up-arrow
7          ;-----
8          ;
9          ; "^" -- Close current location and go to previous one.
10         ;
11 004034  005037  0000000  CMDUP: CLR          D. PCOL          ;Say print column = 0
12 004040  005737  0000000          TST          D. LOC          ;Is a cell open?
13 004044  001002          BNE          3$          ;Br if yes
14 004046  000137  001476'  JMP          NEWLIN        ;Ignore if no cell open
15 004052  105737  0000000  3$: TSTB         D. VIFL        ;Was a new value specified for this cell?
16 004056  003404          BLE          1$          ;Br if not
17 004060  013700  0000000  MOV          D. VAL1, R0    ;Get new value
18 004064  004737  006412'  CALL         STRVAL        ;Store into currently open cell
19 004070  105737  0000000  1$: TSTB         D. BYTM        ;Are we in byte or word mode?
20 004074  001017          BNE          2$          ;Br if in byte mode
21 004076  162737  000002  0000000  SUB          #2, D. LOC     ;Move to previous word
22 004104          TPRINT        #CRLF        ;Go to new line
23 004114  004737  006750'  CALL         SHOLOC        ;Display address of new location
24 004120          PRINT         < / >        ;Display "/"
25 004130  000137  001654'  JMP          CMDSL1        ;Simulate "/" command
26 004134  005337  0000000  2$: DEC          D. LOC          ;Move to previous byte
27 004140          TPRINT        #CRLF        ;go to new line
28 004150  004737  006750'  CALL         SHOLOC        ;Display address of new location
29 004154          PRINT         < \ >        ;Display "\"
30 004164  000137  001732'  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 004170  010046  DBGINI: MOV      RO, -(SP)
8          ;
9          ;  Initialize some debugger values
10         ;
11 004172  012700  0000000  MOV      #D.START, RO      ;Point to start of debugging data
12 004176  005020          1$: CLR      (RO)+      ;Clear all debugging data to 0
13 004200  020027  0000000  CMP      RO, #D.END      ;Finished?
14 004204  103774          BLO      1$          ;Loop if not
15 004206  012737  177777  0000000  MOV      #177777, D.MASK ;Initialize the data mask
16 004214  052737  0000000  0000000  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 004222  052737  0000000  0000000  BIS      #UPMODE, @#PSW ;Say previous mode = user
22         ;
23         ;  Print greeting message
24         ;
25 004230          TPRINT  #TM$GRT      ;Print greeting message
26         ;
27         ;  Finished
28         ;
29 004240  012600          MOV      (SP)+, RO
30 004242  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 004244 010146 BRKRST: MOV      R1,-(SP)
8 004246 032737 0000000 0000000 BIT      #D#BKST,D.FLAG ;Are breakpoints set in program?
9 004254 001427 BEQ      9$          ;Br if not
10 004256 032737 0000000 0000000 BIT      #D#CKBK,D.FLAG ;Are there any active instruction breaks?
11 004264 001427 BEQ      9$          ;Br if not
12          ;
13          ; Reset instructions that were overwritten by breakpoints
14          ;
15 004266 005001 CLR      R1          ;Init index into breakpoint tables
16 004270 016100 0000000 1$: MOV     D.BKAD(R1),R0 ;Get address of breakpoint
17 004274 001407 BEQ      2$          ;Br if this breakpoint not in use
18 004276 106510 MFPD    (R0)         ;Get breakpoint instruction
19 004300 022627 0000003 CMP      (SP)+,#3     ;Is this a breakpoint instruction?
20 004304 001003 BNE     2$          ;Br if not
21 004306 016146 0000000 MOV     D.BKSV(R1),-(SP);Get saved instruction
22 004312 106610 MTPD    (R0)         ;Restore saved instruction
23 004314 062701 0000002 2$: ADD     #2,R1      ;Advance breakpoint index
24 004320 020127 0000020 CMP      R1,#16     ;Checked all breakpoints?
25 004324 101761 BLOS    1$          ;Loop if more to check
26 004326 042737 0000000 0000000 BIC     #D#BKST,D.FLAG ;Say breakpoints no longer set
27          ;
28          ; Finished
29          ;
30 004334 012601 9$: MOV     (SP)+,R1
31 004336 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 004340 010146 BRKSET: MOV      R1,-(SP)
7 004342 032737 0000000 0000000 BIT      #D#CKBK,D.FLAG ;Are there any instruction breakpoints?
8 004350 001416 BEQ      3$          ;Br if not
9                                     ;
10                                    ; Set breakpoints
11                                    ;
12 004352 012701 000020          MOV      #16,R1          ;Get index for last breakpoint
13 004356 016100 0000000 1$: MOV      D.BKAD(R1),R0 ;Get address where breakpoint goes
14 004362 001406 BEQ      2$          ;Br if this breakpoint not in use
15 004364 106510 MFPD     (R0)          ;Get current contents of location
16 004366 012661 0000000 MOV      (SP)+,D.BKSV(R1); Save original contents of cell
17 004372 012746 0000003 MOV      #3,-(SP)      ;Push BPT instruction
18 004376 106610 MTPD     (R0)          ;Store BPT instruction into user's program
19 004400 162701 0000002 2$: SUB      #2,R1          ;More breakpoints to check?
20 004404 002364 BGE      1$          ;Br if yes
21 004406 052737 0000000 0000000 3$: BIS      #D#BKST,D.FLAG ;Set flag saying breakpoints set in program
22                                    ;
23                                    ; Finished
24                                    ;
25 004414 012601 MOV      (SP)+,R1
26 004416 000207 RETURN

```


BRKCHK --- Determine type of breakpoint

```

1          .SBTTL  BRKCHK -- Determine type of breakpoint
2
3          ;-----
4          ; Determine the type of breakpoint that has occurred.
5          ; There are four different types of breakpoints:
6          ; 1. Data breakpoints that are triggered when a monitored cell changes
7          ;    value or takes on a specified value.
8          ; 2. Instruction breakpoints set by use of the "a;nB" instruction.
9          ; 3. Instruction breakpoint used to catch a CALL return.
10         ; 4. Single-step breakpoints that result from the use of the ";IS"
11         ;    command.
12         ;
13         ; Inputs:
14         ; D.R7 = PC after the break
15         ;
16         ; Outputs:
17         ; D#DBRK in D.FLAG ==> Data breakpoint
18         ; D#IBRK in D.FLAG ==> Instruction breakpoint
19         ; D#SBRK in D.FLAG ==> Single-step breakpoint or CALL breakpoint
20         ; D#FBRK in D.FLAG ==> Forced breakpoint (user typed ctrl-B)
21         ; D.BKNM = Breakpoint number
22 004420 010146 BRKCHK: MOV     R1, -(SP)
23
24         ; Clear instruction and data breakpoint flags
25
26 004422 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 004430 032737 000000G 000000G      BIT     #D#DMON, D.FLAG ;Is a data breakpoint active?
31 004436 001425      BEQ     1$ ;Br if not
32 004440 013701 000000G      MOV     D.DADR, R1 ;Get address of monitored cell
33 004444 106511      MFPD   (R1) ;Get contents of monitored cell
34 004446 013701 000000G      MOV     D.MASK, R1 ;Get data mask
35 004452 005101      COM     R1 ;Complement
36 004454 040116      BIC     R1, (SP) ;Leave only bits of interest
37 004456 032737 000000G 000000G      BIT     #D#DVAL, D.FLAG ;Are we waiting for a specified value?
38 004464 001004      BNE     2$ ;Br if yes
39         ; Trigger data breakpoint whenever monitored data cell changes value
40 004466 022637 000000G      CMP     (SP)+, D.DOLD ;Has value changed?
41 004472 001004      BNE     3$ ;Br if yes -- trigger breakpoint
42 004474 000406      BR     1$ ;Hasn't changed -- no data break
43         ; Trigger data breakpoint if value takes on specified value
44 004476 022637 000000G 2$:      CMP     (SP)+, D.DTRG ;Does cell have value of interest?
45 004502 001003      BNE     1$ ;Br if not -- no data breakpoint
46
47         ; Trigger a data breakpoint
48
49 004504 052737 000000G 000000G 3$:      BIS     #D#DBRK, D.FLAG ;Set data-breakpoint flag
50
51         ; See if we need to check for instruction or single-step breakpoints
52
53 004512 032737 000000C 000000G 1$:      BIT     #<D#SSTP!D#TSTP!D#CKBK>, D.FLAG ;Any instruction breaks or stp?
54 004520 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.

```

```

58 ;
59 004522 032737 0000000 0000000 BIT #D$IPND,D.FLAG ;Did we just execute a pending instruction?
60 004530 001036 BNE 9$ ;Br if yes
61 ;
62 ; See if this is a single-step breakpoint
63 ;
64 004532 032737 0000000 0000000 BIT #D$SSTP!D$TSTP,D.FLAG ;Are we single stepping program?
65 004540 001407 BEQ 4$ ;Br if not
66 004542 052737 0000000 0000000 5$: BIS #D$SBRK,D.FLAG ;Set single-step breakpoint flag
67 004550 112737 000011 0000000 MOVB #9.,D.BKNM ;Say this is breakpoint # 9
68 004556 000423 BR 9$
69 ;
70 ; See if this is an instruction breakpoint
71 ;
72 004560 012701 000020 4$: MOV #16.,R1 ;Init breakpoint index
73 004564 013700 0000000 MOV D.R7,R0 ;Get address after breakpoint
74 004570 162700 000002 SUB #2,R0 ;Point to break instruction
75 004574 020061 0000000 7$: CMP R0,D.BKAD(R1) ;Search for which breakpoint was hit
76 004600 001404 BEQ 6$ ;Br if found it
77 004602 162701 000002 SUB #2,R1 ;Get next breakpoint index
78 004606 002372 BGE 7$ ;Loop if more breakpoint to check
79 004610 000406 BR 9$ ;This is not an instruction breakpoint
80 ;
81 ; We hit an instruction breakpoint
82 ;
83 004612 006201 6$: ASR R1 ;Get breakpoint number
84 004614 110137 0000000 MOVB R1,D.BKNM
85 004620 052737 0000000 0000000 BIS #D$IBRK,D.FLAG ;Remember we had an instruction break
86 ;
87 ; Finished
88 ;
89 004626 012601 9$: MOV (SP)+,R1
90 004630 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 004632 010146 SHOBRK: MOV     R1, -(SP)
7 004634          TPRINT  #CRLF          ;Get to start of new line
8          ;
9          ; See if a data breakpoint occurred
10         ;
11 004644 032737 0000000 0000000 BIT     #D$DBRK, D. FLAG ;Did a data breakpoint occur?
12 004652 001454          BEQ     1$          ;Br if not
13 004654          PRINT   <Data break at >
14 004700 013700 0000000          MOV     D, R7, RO          ;Get breakpoint PC
15 004704 004737 007146'          CALL   LSTADR          ;Display the address
16 004710          PRINT   < -- >
17 004722 013700 0000000          MOV     D, DADR, RO          ;Get address of monitored cell
18 004726 004737 007146'          CALL   LSTADR          ;Display it
19 004732          PRINT   < = >
20 004742 013700 0000000          MOV     D, DADR, RO          ;Get address of cell
21 004746 106510          MFPD   (RO)          ;Get current contents of cell
22 004750 012600          MOV     (SP)+, RO
23 004752 010037 0000000          MOV     RO, D, DOLD          ;Save old cell value
24 004756 004737 006100'          CALL   PRTWRD          ;Display value of cell
25 004762 013700 0000000          MOV     D, MASK, RO          ;Get data mask
26 004766 005100          COM     RO          ;Complement
27 004770 040037 0000000          BIC     RO, D, DOLD          ;Clear all but bits of interest
28 004774          TPRINT  #CRLF          ;Go to new line
29         ;
30         ; See if a single-step breakpoint occurred
31         ;
32 005004 032737 0000000 0000000 1$: BIT     #D$SBRK, D. FLAG ;Did a single-step breakpoint occur?
33 005012 001406          BEQ     2$          ;Br if not
34 005014          5$: PRINT   <Step: >          ;Print heading
35 005026 000451          BR     3$          ;Now treat like instruction breakpoing
36         ;
37         ; See if an instruction breakpoint occurred
38         ;
39 005030 032737 0000000 0000000 2$: BIT     #D$IBRK, D. FLAG ;Instruction breakpoint?
40 005036 001426          BEQ     4$          ;Br if not
41 005040          TPRINT  #CRLF          ;Go to new line
42 005050 123727 0000000 000010          CMPB   D, BKNM, #8.          ;CALL breakpoint?
43 005056 001003          BNE     6$          ;Br if not
44 005060 005037 0000000          CLR     D, CBRK          ;Clear the call breakpoint
45 005064 000753          BR     5$          ;Br if yes -- Treat like single step break
46 005066          6$: PRINT   <B>
47 005074 113700 0000000          MOVB   D, BKNM, RO          ;Get breakpoint number
48 005100 004737 005772'          CALL   PRTOCT          ;Print breakpoint number
49 005104          PRINT   <: >
50 005112 000417          BR     3$
51         ;
52         ; See if we had a forced breakpoint
53         ;
54 005114 032737 0000000 0000000 4$: BIT     #D$FBRK, D. FLAG ;Forced breakpoint?
55 005122 001427          BEQ     9$          ;Br if not
56 005124          TPRINT  #CRLF          ;Go to a new line
57 005134          PRINT   <Break at >

```

```
58 ;  
59 ; Display the address of the breakpoint  
60 ;  
61 005152 013700 0000000 3$: MOV D,R7,R0 ;Get address where break occurred  
62 005156 004737 007146' CALL LSTADR ;Display the address  
63 ;  
64 ; Display the decoded instruction  
65 ;  
66 005162 013701 0000000 MOV D,R7,R1 ;Get address of break  
67 005166 004737 007510' CALL DECODE ;Display the decoded instruction  
68 005172 TPRINT #CRLF ;Go to new line  
69 ;  
70 ; Finished  
71 ;  
72 005202 012601 9$: MOV (SP)+,R1  
73 005204 000207 RETURN
```

FLGBRK -- Set flag if any breakpoints are active

```

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 005206  010146  FLGBRK: MOV      R1,-(SP)
8          ;
9          ; Determine if there are any normal instruction breakpoints
10         ;
11 005210  012701  000020          MOV      #16,R1          ;Get index to last breakpoint entry
12 005214  005761  000000G 5$:  TST      D.BKAD(R1)      ;Is this breakpoint set?
13 005220  001007          BNE      6$          ;Br if yes
14 005222  162701  000002          SUB      #2,R1          ;More to check?
15 005226  002372          BGE      5$          ;Loop if yes
16         ;
17         ; No breakpoints are active
18         ;
19 005230  042737  000000G 000000G          BIC      #D$CKBK,D.FLAG ;No breakpoints are set
20 005236  000403          BR       9$
21         ;
22         ; There are active breakpoints
23         ;
24 005240  052737  000000G 000000G 6$:  BIS      #D$CKBK,D.FLAG ;Remember some breakpoint is set
25         ;
26         ; Finished
27         ;
28 005246  012601  9$:  MOV      (SP)+,R1
29 005250  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 $x
5          ; where "r" is a relocation base register number,
6          ; and "x" is an internal register number.
7          ;
8          ; Outputs:
9          ; R0 = Value accrued.
10         ; R1 = Status flag:
11         ; +1 ==> normal value
12         ; 0 ==> No value gotten.
13         ; -1 ==> Internal register address.
14         ;
15 005252 010246 ACRVAL: MOV      R2,-(SP)
16         ;
17         ; Get 1st character and see if this is a normal value or a register name.
18         ;
19 005254 004737 005720' CALL    GETCHR      ;Get 1st character
20 005260 120027 000044  CMPB   RO,#'$      ;Is this a register value?
21 005264 001461          BEQ    1$          ;Branch if yes
22 005266 120027 000056  CMPB   RO,#'.'    ;"." = Current program counter
23 005272 001451          BEQ    10$         ;Br if period
24 005274 004737 005764' CALL    PSHCHR     ;Save the character
25 005300 120027 000053  CMPB   RO,#'+'    ;Is this a sign character?
26 005304 001414          BEQ    2$          ;Br if yes
27 005306 120027 000055  CMPB   RO,#'-'    ;
28 005312 001411          BEQ    2$          ;
29 005314 120027 000060  CMPB   RO,#'0'    ;Is this character a digit?
30 005320 103403          BLD    8$          ;Br if not a digit
31 005322 120027 000071  CMPB   RO,#'9'    ;
32 005326 101403          BLOS  2$          ;Br if a digit
33         ;
34         ; No value was specified
35         ;
36 005330 005001 8$:     CLR    R1          ;Return status code
37 005332 005000          CLR    R0          ;Return value of 0
38 005334 000457          BR     9$          ;
39         ;
40         ; Accrue normal value
41         ;
42 005336 004737 005500' 2$:     CALL   ACRNUM     ;Accrue first part of number
43         ;
44         ; See if 1st part of value is a relocation base register number
45         ;
46 005342 010102          MOV    R1,R2       ;Save value that was accrued
47 005344 120027 000054  CMPB   RO,#','    ;Was first part terminated with a comma?
48 005350 001016          BNE   3$          ;Br if not
49 005352 020227 000007  CMP    R2,#7      ;Is first part a valid register number?
50 005356 101404          BLOS  4$          ;Br if ok
51 005360          ERR    #EM#IRB
52 005370 006302 4$:     ASL    R2          ;Convert reg # to word table index
53 005372 004737 005720' CALL   GETCHR     ;Skip over comma
54 005376 004737 005500' CALL   ACRNUM     ;Accrue second part of number
55 005402 066201 0000000 ADD    D,RLBS(R2),R1 ;Add relocation base to offset value
56 005406 010100 3$:     MOV    R1,R0       ;Return value in R0
57 005410 012701 000001  MOV    #1,R1      ;Return status code in R1

```

```
58 005414 000427          BR      9$          ;Return
59                          ;
60                          ; Value is "." (period). Use current program counter.
61                          ;
62 005416 013700 0000000 10$:   MOV      D, R7, R0      ;Get current program counter value (R7)
63 005422 012701 000001  MOV      #1, R1      ;Say we got a normal value
64 005426 000422          BR      9$          ;Return
65                          ;
66                          ; Internal value ("x")
67                          ;
68 005430 004737 005720' 1$:   CALL     GETCHR      ;Get register letter
69 005434 012701 000014  MOV      #NUMINT-1, R1 ;Get # internal registers
70 005440 120061 000362' 6$:   CMPB    RO, INTCHR(R1) ;Look up register letter
71 005444 001406          BEQ      5$          ;Br if found
72 005446 005301          DEC      R1          ;More to check?
73 005450 002373          BGE      6$          ;Loop if yes
74 005452          ERR      #EM#IIV ;Invalid register name
75 005462 006301 5$:   ASL      R1          ;Convert index to word table index
76 005464 016100 000400' MOV      INTADR(R1), R0 ;Get address of cell for register
77 005470 012701 177777 MOV      #-1, R1      ;Get status code
78                          ;
79                          ; Finished
80                          ;
81 005474 012602 9$:   MOV      (SP)+, R2
82 005476 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         ACRNUM: MOV     R2, -(SP)
11         MOV     R3, -(SP)
12         MOV     R5, -(SP)
13         ;
14         ; See if number has a leading sign character
15         ;
16         CLR     R2          ; Assume result should be positive
17         CALL   GETCHR      ; Get 1st character of number
18         CMPB   R0, #'+'    ; Leading plus sign?
19         BEQ    6$          ; Br if yes
20         CMPB   R0, #'-'    ; Leading minus sign?
21         BNE    8$          ; Br if not
22         INC    R2          ; Set flag to negate the value
23         BR     6$
24         B$:    CALL   PSHCHR ; Push the first digit
25         ;
26         ; Scan the number and store in D.NMBF
27         ;
28         6$:    MOV     #D.NMBF, R1 ; Point to number character buffer
29         MOV     #8, R5       ; Assume this is an octal value
30         1$:    CALL   GETCHR      ; Get next character of number
31         CMPB   R0, #'0'    ; Is this character a digit?
32         BLO    2$          ; Br if not
33         CMPB   R0, #'9'    ; Decimal digit?
34         BHI    2$          ; Br if not
35         CMPB   R0, #'7'    ; Octal digit?
36         BLOS   3$          ; Br if yes
37         MOV     #10, R5     ; This must be a decimal value
38         3$:    CMP     R1, #D.NMBE ; Are we about to overflow the buffer?
39         BHIS   11$         ; Br if yes
40         MOVB   R0, (R1)+    ; Store character into buffer
41         BR     1$          ; Loop to get rest of number
42         ;
43         ; Hit end of number
44         ;
45         2$:    CLRB   (R1)    ; Put null at end of buffer
46         CMPB   R0, #'.'    ; Decimal point at end of number?
47         BNE    4$          ; Br if not
48         MOV     #10, R5     ; Remember this is a decimal value
49         BR     5$
50         4$:    CALL   PSHCHR      ; Save delimiter
51         ;
52         ; Convert number string to binary value
53         ;
54         5$:    MOV     #D.NMBF, R3 ; Point to start of buffer
55         CLR    R1          ; Store value in R1
56         7$:    MOVB   (R3)+, R0 ; Get next character
57         BEQ    9$          ; Br if hit end of number

```



```
58 005650 162700 000060          SUB    #'0,R0          ;Convert digit to binary value
59 005654 070100          MUL    R5,R1          ;Multiply previous value by 8 or 10.
60 005656 060001          ADD    R0,R1          ;Add in new value
61 005660 000771          BR     7#            ;Loop to get rest of number
62                               ;
63                               ; See if we should negate the value
64                               ;
65 005662 005702          9#:   TST    R2          ;Should we negate the value
66 005664 001401          BEQ    10#          ;Br if not
67 005666 005401          NEG    R1          ;Negate the value
68                               ;
69                               ; Finished
70                               ;
71 005670 004737 005720'    10#:  CALL   GETCHR        ;Get delimiter character in R0
72 005674 004737 005764'    CALL   PSHCHR        ;But push it back too
73 005700 012605          MOV    (SP)+,R5
74 005702 012603          MOV    (SP)+,R3
75 005704 012602          MOV    (SP)+,R2
76 005706 000207          RETURN
77                               ;
78                               ; Number overflowed the buffer
79                               ;
80 005710          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 005720  GETCHR:
9          ;
10         ; See if PSHCHR was called to save a character
11         ;
12 005720 113700 0000000  MOVB    D,SVCH,RO      ;Has a character been pushed?
13 005724 001014          BNE     9$              ;Br if yes
14         ;
15         ; No saved character.
16         ; Get character from terminal.
17         ;
18 005726          .TTYIN          ;Get character from terminal
19 005732 005237 0000000  INC     D,PCOL          ;Advance print column
20         ;
21         ; Convert lower-case characters to upper-case
22         ;
23 005736 120027 000141    CMPB    RO,#141         ;Lower-case a
24 005742 103405          BLD     9$              ;
25 005744 120027 000172    CMPB    RO,#172         ;Lower-case z
26 005750 101002          BHI     9$              ;
27 005752 162700 000040    SUB     #40,RO          ;Convert lower-case to upper-case
28         ;
29         ; Finished
30         ;
31 005756 105037 0000000 9$:  CLRB   D,SVCH          ;No saved character
32 005762 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 005764 110037 0000000 PSHCHR: MOVB   RO,D,SVCH      ;Save the character for GETCHR
42 005770 000207          RETURN          ;Finished

```

PRTOCT -- Print octal value

```

1
2
3
4
5
6
7
8 005772 010146
9 005774 010246
10 005776 010346
11 006000 005003
12 006002 010001
13 006004 005000
14 006006 073027 000001
15 006012 012702 000006
16 006016 000403
17 006020 005000
18 006022 073027 000003
19 006026 050003
20 006030 001406
21 006032 062700 000060
22 006036
23 006042 005237 0000006
24 006046 077214
25 006050 005703
26 006052 001006
27 006054
28 006064 005237 0000006
29 006070 012603
30 006072 012602
31 006074 012601
32 006076 000207

```

```

.SBTTL PRTOCT -- Print octal value
-----
; Print an octal value with no leading zeroes.
;
; Inputs:
; R0 = Value to print.
;
PRTOCT: MOV R1, -(SP)
        MOV R2, -(SP)
        MOV R3, -(SP)
        CLR R3 ; Say non-zero digit not seen yet
        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$: CLR R0 ; Get next digit into R0
   ASHC #3, R0
2$: BIS R0, R3 ; Remember if non-zero digit seen
   BEQ 3$ ; Br if no non-zero digits seen yet
   ADD #'0, R0 ; Convert to ascii character
   .TTYOUT ; Print the character
   INC D, PCOL ; Advance print column
3$: SOB R2, 1$ ; Loop if more digits to print
   TST R3 ; Did we see any non-zero digits?
   BNE 4$ ; Br if yes
   .TTYOUT #'0 ; Print one zero if not
   INC D, PCOL ; Advance print column
4$: MOV (SP)+, R3
   MOV (SP)+, R2
   MOV (SP)+, R1
   RETURN

```

PRTWRD -- Print octal word value

```

1
2
3
4
5
6
7 006100 010146
8 006102 010246
9 006104 010001
10 006106 005000
11 006110 073027 000001
12 006114 012702 000006
13 006120 000403
14 006122 005000
15 006124 073027 000003
16 006130 062700 000060
17 006134
18 006140 005237 0000006
19 006144 077212
20 006146 012602
21 006150 012601
22 006152 000207
23
24
25
26
27
28
29
30
31 006154 010146
32 006156 010246
33 006160 010001
34 006162 000301
35 006164 005000
36 006166 073027 000002
37 006172 012702 000003
38 006176 000403
39 006200 005000
40 006202 073027 000003
41 006206 062700 000060
42 006212
43 006216 005237 0000006
44 006222 077212
45 006224 012602
46 006226 012601
47 006230 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

```

PRTR50 -- Print a RAD50 value

```

1          .SBTTL  PRTR50 -- Print a RAD50 value
2          ;-----
3          ; PRTR50 is called to print a RAD50 (radix 50) value.
4          ;
5          ; Inputs:
6          ; R0 = Value to be printed.
7          ;
8 006232 010146 PRTR50: MOV     R1,-(SP)
9 006234 010246          MOV     R2,-(SP)
10         ;
11         ; Convert value to ascii character string and stack the characters
12         ;
13 006236 012702 000003          MOV     #3,R2          ;Get # chars to convert
14 006242 005046          CLR     -(SP)          ;Put null on stack to signal end
15 006244 010001          MOV     R0,R1          ;Get value to be converted
16 006246 005000 1$: CLR     R0          ;Clear high-order for divide
17 006250 071027 000050          DIV     #50,R0          ;Divide R0-R1 by 50
18 006254 116146 000220'          MOVB   R50CHR(R1),-(SP);Convert remainder to ascii and stack it
19 006260 010001          MOV     R0,R1          ;Get quotient
20 006262 077207          SOB     R2,1$          ;Loop if more to convert
21         ;
22         ; Finished conversion. Print the result.
23         ;
24 006264 012600 2$: MOV     (SP)+,R0          ;Get char to print
25 006266 001405          BEQ     3$          ;Br if hit end of string
26 006270          .TTYOUT          ;Print the character
27 006274 005237 0000006          INC     D.PCOL          ;Advance print column
28 006300 000771          BR     2$          ;Loop to print more
29         ;
30         ; Finished
31         ;
32 006302 012602 3$: MOV     (SP)+,R2
33 006304 012601          MOV     (SP)+,R1
34 006306 000207          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 006310  ERRPRT: TPRINT  #CRLF          ;Go to new line
7 006320          TPRINT  R1           ;Print error message
8 006326  005037  0000000          CLR      D.PCOL          ;Say print column = 0
9 006332  013704  0000000          MOV      D.SPSV,SP       ;Reset stack pointer
10 006336  000137  001506'          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 006342  112200  LSTTXT: MOVB      (R2)+,R0       ;Get next char from text string
21 006344  001405          BEQ      2$           ;Br if hit end of string
22 006346          .TTYOUT          ;Print the character
23 006352  005237  0000000          INC      D.PCOL          ;Advance print column
24 006356  000771          BR       LSTTXT          ;Loop till end of string hit
25 006360  005202          2$: INC      R2           ;Bound up to even byte address
26 006362  042702  000001          BIC      #1,R2
27 006366  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 006370 113700 0000000 SETLOC: MOVB   D.V1FL,RO    ;Get mode of address
14 006374 001405         BEQ     9$           ;Br if no address specified
15 006376 110037 0000000         MOVB   RO,D.LOCM   ;Set mode of current cell
16 006402 013737 0000000 0000000 MOV    D.VAL1,D.LOC  ;Set address of current cell
17 006410 000207         9$:   RETURN

```

STRVAL -- Store value into open cell

```

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 006412  010146  STRVAL: MOV      R1,-(SP)
12 006414  010246          MOV      R2,-(SP)
13         ;
14         ; Make sure a cell is open
15         ;
16 006416  105737  0000000  TSTB   D.LOCM          ;Is a cell open now?
17 006422  001471          BEQ     9$                ;Br if not
18         ;
19         ; Determine if we are storing into user's job or internal cell
20         ;
21 006424  105737  0000000  TSTB   D.LOCM          ;Internal or user job?
22 006430  100436          BMI     1$                ;Br if internal cell
23         ;
24         ; Store into user's job
25         ;
26 006432  105737  0000000  TSTB   D.BYTM          ;Byte or word mode?
27 006436  001005          BNE     2$                ;Br if byte mode
28         ;
29         ; Store into word
30         ;
31 006440  010046          MOV     RO,-(SP)        ;Put value on stack
32 006442  013700  0000000  MOV     D.LOC,R0       ;Get address where we should store
33 006446  106610          MTPD   (R0)           ;Store into user's area
34 006450  000456          BR     9$
35         ;
36         ; Store into byte
37         ;
38 006452  042700  177400  2$:    BIC     #^C377,R0   ;Clear high-order byte of value being stored
39 006456  013702  0000000  MOV     D.LOC,R2       ;Get address where we are to store
40 006462  042702  0000001  BIC     #1,R2          ;Force address to be even
41 006466  106512          MFPD   (R2)           ;Get current contents of word
42 006470  032737  0000001  0000000  BIT     #1,D.LOC       ;Even or odd byte?
43 006476  001005          BNE     3$                ;Br if odd byte
44 006500  042716  000377  BIC     #377,(SP)      ;Clear low-order byte of old value
45 006504  050016          BIS     RO,(SP)        ;Put in new low-order byte
46 006506  106612          MTPD   (R2)           ;Store new word
47 006510  000436          BR     9$
48 006512  042716  177400  3$:    BIC     #177400,(SP) ;Clear high-order byte
49 006516  000300          SWAB   R0              ;Get new value to high-order byte
50 006520  050016          BIS     RO,(SP)        ;Store new high-order byte
51 006522  106612          MTPD   (R2)           ;Store new word
52 006524  000430          BR     9$
53         ;
54         ; Store into internal register
55         ;
56 006526  105737  0000000  1$:    TSTB   D.BYTM          ;Byte or word mode?
57 006532  001003          BNE     12$             ;Br if byte mode

```


STRVAL -- Store value into open cell

```

58 ;
59 ; Store into word
60 ;
61 006534 010077 0000000 MOV R0,@D.LOC ;Store into internal cell
62 006540 000422 BR 9#
63 ;
64 ; Store into byte
65 ;
66 006542 042700 177400 12#: BIC #^C377,R0 ;Clear high-order byte of value being stored
67 006546 013702 0000000 MOV D.LOC,R2 ;Get address where we are to store
68 006552 042702 000001 BIC #1,R2 ;Force address to be even
69 006556 032737 000001 0000000 BIT #1,D.LOC ;Even or odd byte?
70 006564 001352 BNE 3# ;Br if odd byte
71 006566 042712 000377 BIC #377,(R2) ;Clear low-order byte of old value
72 006572 050012 BIS R0,(R2) ;Put in new low-order byte
73 006574 000404 BR 9#
74 006576 042712 177400 13#: BIC #177400,(R2) ;Clear high-order byte
75 006602 000300 SWAB R0 ;Get new value to high-order byte
76 006604 050012 BIS R0,(R2) ;Store new high-order byte
77 ;
78 ; Finished
79 ;
80 006606 012602 9#: MOV (SP)+,R2
81 006610 012601 MOV (SP)+,R1
82 006612 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 006614 010246 GETVAL: MOV      R2,-(SP)
14         ;
15         ; Determine if we are fetching from user's job or internal cell
16         ;
17 006616 105737 0000000 TSTB   D.LOCM      ;Internal or user job?
18 006622 100426        BMI    1$          ;Br if internal cell
19         ;
20         ; Fetch from user's job
21         ;
22 006624 105737 0000000 TSTB   D.BYTM      ;Byte or word mode?
23 006630 001005        BNE    2$          ;Br if byte mode
24         ;
25         ; Fetch from word
26         ;
27 006632 013700 0000000 MOV     D.LOC,R0   ;Get address of value wanted
28 006636 106510        MFPD   (R0)        ;Get value from user's area
29 006640 012600        MOV    (SP)+,R0    ;Return in RO
30 006642 000440        BR     9$
31         ;
32         ; Fetch from byte
33         ;
34 006644 013702 0000000 2$:  MOV     D.LOC,R2   ;Get address of word
35 006650 042702 0000001    BIC    #1,R2      ;Force address to be even
36 006654 106512        MFPD   (R2)        ;Get current contents of word
37 006656 012600        MOV    (SP)+,R0    ;Get word value to RO
38 006660 032737 0000001 0000000 BIT     #1,D.LOC   ;Even or odd byte?
39 006666 001401        BEQ    3$          ;Br if low-order byte wanted
40 006670 000300        SWAB   RO          ;Swap bytes if high-order byte wanted
41 006672 042700 177400 3$:  BIC    #177400,R0  ;Clear high-order byte
42 006676 000422        BR     9$
43         ;
44         ; Fetch from internal register
45         ;
46 006700 105737 0000000 1$:  TSTB   D.BYTM      ;Byte or word mode?
47 006704 001003        BNE    12$         ;Br if byte mode
48         ;
49         ; Fetch from word
50         ;
51 006706 017700 0000000        MOV     @D.LOC,R0   ;Get the value
52 006712 000414        BR     9$
53         ;
54         ; Fetch from byte
55         ;
56 006714 013702 0000000 12$: MOV     D.LOC,R2   ;Get address where we are to fetch
57 006720 042702 0000001    BIC    #1,R2      ;Force address to be even

```

GETVAL -- Get value from current cell

```
58 006724 011200          MOV      (R2),R0          ;Get the word value
59 006726 032737 000001 0000006  BIT      #1,D.LDC        ;Even or odd byte?
60 006734 001401          BEQ      13$              ;Br if even byte wanted
61 006736 000300          SWAB     R0              ;Swap high-order byte to low-order
62 006740 042700 177400    13$:  BIC      #177400,R0      ;Clear high-order byte
63                                     ;
64                                     ; Finished
65                                     ;
66 006744 012602          9$:  MOV      (SP)+,R2
67 006746 000207          RETURN
```

```

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 006750  010146 SHOLOC: MOV     R1,-(SP)
12 006752  010246      MOV     R2,-(SP)
13 006754  010346      MOV     R3,-(SP)
14 006756  010446      MOV     R4,-(SP)
15         ;
16         ; Determine if this is an internal or external cell
17         ;
18 006760  105737  0000000  TSTB   D.LOCM      ;Internal or external address?
19 006764  002405      BLT     1$          ;Br if internal address
20         ;
21         ; Display address of cell in user's program
22         ;
23 006766  013700  0000000  MOV     D.LOC,R0    ;Get address to be displayed
24 006772  004737  007146'  CALL   LSTADR      ;Display relative to relocation base
25 006776  000456      BR      9$
26         ;
27         ; Display name of internal cell
28         ;
29 007000  013700  0000000  1$:    MOV     D.LOC,R0    ;Get address of internal cell
30 007004  012701  000400'  MOV     #INTADR,R1  ;Point to table of addresses of cells
31 007010  012702  177777  MOV     #-1,R2      ;Init for search
32 007014  005003      CLR     R3
33 007016  020011  2$:    CMP     R0,(R1)      ;Is address below base of this cell?
34 007020  103406      BLO    3$          ;Br if yes
35 007022  010004      MOV     R0,R4      ;Get address of interest
36 007024  161104      SUB     (R1),R4     ;Determine distance from cell base addr
37 007026  020402      CMP     R4,R2      ;Is this the closest we've seen so far?
38 007030  101002      BHI    3$          ;Br if not
39 007032  010402      MOV     R4,R2      ;Remember offset from nearest cell
40 007034  010103      MOV     R1,R3      ;Remember address of nearest cell
41 007036  062701  000002  3$:    ADD     #2,R1      ;Point to address of next cell
42 007042  020127  000432'  CMP     R1,#INTEND  ;Checked all cells?
43 007046  103763      BLO    2$          ;Loop if not
44         ;
45         ; Display register name
46         ;
47 007050  162703  000400'  SUB     #INTADR,R3  ;Get offset into address table
48 007054  006203      ASR     R3          ;Convert to byte table index
49 007056      .TTYOUT #'$      ;Print "$"
50 007066  005237  0000000  INC     D.PCOL      ;Advance print column
51 007072  116300  000362'  MOVB   INTCHR(R3),R0 ;Get name of internal register
52 007076      .TTYOUT          ;Print register name
53 007102  005237  0000000  INC     D.PCOL      ;Advance print column
54         ;
55         ; Display offset from register
56         ;
57 007106  005702      TST     R2          ;Any offset from base of cell?

```

```
58 007110 001411          BEQ      9$          ;Br if not
59 007112                .TTYOUT #'+'      ;Print "+"
60 007122 005237 0000000  INC      D.PCOL      ;Advance print column
61 007126 010200          MOV      R2,R0       ;Get offset value
62 007130 004737 005772'  CALL     PRTOCT      ;Print octal value for offset
63
64                ; Finished
65
66 007134 012604          9$:    MOV      (SP)+,R4
67 007136 012603          MOV      (SP)+,R3
68 007140 012602          MOV      (SP)+,R2
69 007142 012601          MOV      (SP)+,R1
70 007144 000207          RETURN
```

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 007146  LSTADR:
13         ;
14         ; See if DP$LAA is set in $F register
15         ;
16 007146  032737  0000000 0000000  BIT    #DP$LAA,D.PFMT ;Absolute addresses wanted?
17 007154  001003          1$      ;Br if yes
18         ;
19         ; Display address with relocation base
20         ;
21 007156  004737  007172'  CALL   SHOADR      ;Display address with relocation base
22 007162  000402          BR     9$
23         ;
24         ; Display absolute address
25         ;
26 007164  004737  005772'  1$:   CALL   PRTOCT      ;Display absolute address
27         ;
28         ; Finished
29         ;
30 007170  000207          9$:   RETURN

```

```

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 007262 010146 RELADR: MOV     R1,-(SP)
15 007264 010246         MOV     R2,-(SP)
16 007266 010446         MOV     R4,-(SP)
17         ;
18         ; Search for relocation region defined below the address
19         ;
20         CLR     R1          ; Init relocation register index
21 007272 012702 177777   MOV     #-1,R2          ; Init closest offset value
22 007276 010203         MOV     R2,R3          ; Say no relocation base found yet
23 007300 005761 000000G 2$:   TST     D,RLBS(R1)    ; Is this relocation register in use?
24 007304 001412         BEQ     3$          ; Br if not
25 007306 020061 000000G   CMP     R0,D,RLBS(R1)  ; Is address above this relocation base?
26 007312 103407         BLO     3$          ; Br if not
27 007314 010004         MOV     R0,R4          ; Get address of interest
28 007316 166104 000000G   SUB     D,RLBS(R1),R4  ; Calculate offset within region
29 007322 020402         CMP     R4,R2          ; Is this the closest one so far?
30 007324 101002         BHI     3$          ; Br if not
31 007326 010402         MOV     R4,R2          ; Rember smallest offset
32 007330 010103         MOV     R1,R3          ; Remember relocation register index
33 007332 062701 000002G 3$:   ADD     #2,R1          ; Advance relocation register index
34 007336 020127 000016   CMP     R1,#14        ; Checked all relocation registers?
35 007342 101756         BLOS   2$          ; Br if not
36         ;
37         ; See if we found a relocation region
38         ;
39 007344 005703         TST     R3          ; Did we find a relocation region
40 007346 002002         BGE     5$          ; Br if yes
41 007350 000261         SEC          ; Say no region found
42 007352 000401         BR     9$          ;
43 007354 000241 5$:   CLC          ; Say a region was found
44         ;
45         ; Finished
46         ;
47 007356 012604 9$:   MOV     (SP)+,R4
48 007360 012602         MOV     (SP)+,R2
49 007362 012601         MOV     (SP)+,R1
50 007364 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 007366 010146 DOPRT:  MOV     R1, -(SP)
10 007370 010246      MOV     R2, -(SP)
11          ;
12          ; Set up pointer to start of string
13          ;
14 007372 010002      MOV     RO, R2          ;Get pointer to start of string
15 007374 113701 0000000 MOVB   CORUSR, R1      ;Get current job index number
16          ;
17          ; Get each char out of the string and print it
18          ;
19 007400 112200 1$:   MOVB   (R2)+, RO      ;Get next char from string
20 007402 001411      BEQ     2$          ;Br if hit null at end
21 007404 120027 000200  CMPB   RO, #200        ;Is string terminated with a 200?
22 007410 001422      BEQ     9$          ;Br if yes
23 007412      DCALL  PUTCHR      ;Print the character
24 007420 005237 0000000 INC     D.PCOL        ;Advance print column
25 007424 000765      BR      1$          ;Loop to print rest of string
26          ;
27          ; Print Carriage-return Line-feed at end of string
28          ;
29 007426 012700 000015 2$:   MOV     #CR, RO      ;Get carriage return
30 007432      DCALL  PUTCHR      ;Print it
31 007440 012700 000012  MOV     #LF, RO      ;Get line feed
32 007444      DCALL  PUTCHR      ;Print it
33 007452 005037 0000000 CLR     D.PCOL        ;Say we are back to column 1
34          ;
35          ; Finished
36          ;
37 007456 012602 9$:   MOV     (SP)+, R2
38 007460 012601      MOV     (SP)+, R1
39 007462 000207      RETURN

```

DBGON -- Place terminal in debug mode

```

1          .SBTTL  DBGON  -- Place terminal in debug mode
2          ;-----
3          ;  DBGON is called to place the terminal in debug mode.
4          ;
5 007464  012700  000002'  DBGON:  MOV    #ONEMT,RO
6 007470  104375          EMT    375
7 007472  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 007474  012700  000004'  DBGOFF: MOV    #OFFEMT,RO
14 007500  104375          EMT    375
15 007502  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 007504  000241  CHKADR: CLC
30 007506  000207          RETURN

```

DECODE -- Decode instruction into symbolic form

```

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 007510 010146 DECODE: MOV     R1,-(SP)
13 007512 010246          MOV     R2,-(SP)
14 007514 010346          MOV     R3,-(SP)
15 007516 010446          MOV     R4,-(SP)
16 007520 010546          MOV     R5,-(SP)
17         ;
18         ; Initialize instruction length to 2 bytes
19         ;
20 007522 012737 000002 0000006      MOV     #2,D.ILEN      ; Say base instruction length = 2 bytes
21         ;
22         ; Initialize column counter for decoded instruction display
23         ;
24 007530          PRINT    <  >      ; First print a few spaces
25 007542 005037 0000006      CLR     D.PCOL        ; Say we are at column 1 now
26         ;
27         ; Fetch the instruction word
28         ;
29 007546 106521          MFPD    (R1)+      ; Get the instruction from user's space
30 007550 012602          MOV     (SP)+,R2      ; Get instruction off of stack
31         ;
32         ; Search the instruction list for this instruction
33         ;
34 007552 012704 011764'      MOV     #INSBAS,R4      ; Point to base of instruction decode table
35 007556 010203 1$:      MOV     R2,R3          ; Get instruction to be decoded
36 007560 046403 000000      BIC     IB$MSK(R4),R3   ; Clear operand fields in instruction
37 007564 020364 000002      CMP     R3,IB$VAL(R4)   ; Is this the instruction?
38 007570 001411          BEQ     5$            ; Br if found the instruction
39 007572 116400 000005      MOVVB  IB$LEN(R4),R0    ; Get length of ascii name string
40 007576 005200          INC     R0            ; Bound up to word boundary
41 007600 042700 000001      BIC     #1,R0          ;
42 007604 062700 000006      ADD     #IB$NAM,R0      ; Add size of base portion of block
43 007610 060004          ADD     R0,R4          ; Point to next instruction decode block
44 007612 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 007614 116403 000005 5$:      MOVVB  IB$LEN(R4),R3    ; Get length of instruction mnemonic
51 007620 060337 0000006      ADD     R3,D.PCOL      ; Advance column counter
52 007624 010405          MOV     R4,R5          ; Point to the mnemonic ascii string
53 007626 062705 000006      ADD     #IB$NAM,R5      ;
54 007632 112500 6$:      MOVVB  (R5)+,R0        ; Get next char of mnemonic
55 007634          .TTYOUT          ; Print the character
56 007640 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 007642 016400 000000      MOV      IB#MSK(R4),R0      ;Get mask value
62 007646 005100      COM      R0      ;Complement it to leave the operand fields
63 007650 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 007652 116400 000004      MOVVB   IB#TYP(R4),R0      ;Get the operand type code for this inst
71 007656 004770 014200'    CALL   @TYPVEC(R0)      ;Call appropriate operand display routine
72      ;
73      ; Finished
74      ;
75 007662 012605      MOV      (SP)+,R5
76 007664 012604      MOV      (SP)+,R4
77 007666 012603      MOV      (SP)+,R3
78 007670 012602      MOV      (SP)+,R2
79 007672 012601      MOV      (SP)+,R1
80 007674 000207      RETURN

```

ODCxxx -- Display instruction operands

```

1          .SBTTL  ODCxxx -- Display instruction operands
2          ;-----
3          ; TYP1 -- Instruction has no operands.
4          ;
5 007676  000207      ODC1:  RETURN
6          ;
7          ;-----
8          ; TYP2 -- Operand has a single general operand.
9          ;
10 007700      ODC2:
11          ;
12          ; Tab over to operand field
13          ;
14 007700  012700  000010      MOV      #COLOPN,RO      ;Get operand column number
15 007704  004737  011734'     CALL      ODCTAB      ;Tab over to operand field
16          ;
17          ; Display the operand
18          ;
19 007710  010200      MOV      R2,RO      ;Get the operand code
20 007712  004737  010650'     CALL      ODCGEN      ;Display general operand
21          ;
22          ; Finished
23          ;
24 007716  000207      RETURN
25          ;
26          ;-----
27          ; TYP3 -- Single register operand.
28          ;
29 007720      ODC3:
30          ;
31          ; Tab over to the operand field
32          ;
33 007720  012700  000010      MOV      #COLOPN,RO      ;Get operand column number
34 007724  004737  011734'     CALL      ODCTAB      ;Tab over to operand field
35          ;
36          ; Display the register
37          ;
38 007730  010200      MOV      R2,RO      ;Get the register number
39 007732  004737  011342'     CALL      ODCREG      ;Display the register number
40 007736  000207      RETURN

```

ODCxxx -- Display instruction operands

```

1 ;-----
2 ; TYP4 -- 4-bit absolute numeric operand
3 ;
4 007740 ODC4:
5 ;
6 ; Tab over to the operand field
7 ;
8 007740 012700 000010 MOV #COLPN,R0 ;Get operand column number
9 007744 004737 011734' CALL ODCTAB ;Tab over to operand field
10 ;
11 ; Display the value
12 ;
13 007750 010200 MOV R2,R0 ;Get value to display
14 007752 004737 005772' CALL PRTOCT ;Print the value
15 ;
16 ; Finished
17 ;
18 007756 000207 RETURN
19 ;-----
20 ;
21 ; TYP5 -- Condition code modification instruction
22 ;
23 007760 ODC5:
24 ;
25 ; Determine which condition code is affected
26 ;
27 007760 032702 000001 BIT #1,R2 ;C-flag?
28 007764 001404 BEQ 1$ ;Br if not
29 007766 .TTYOUT #'C
30 007776 032702 000002 1$: BIT #2,R2 ;V-flag?
31 010002 001404 BEQ 2$ ;Br if not
32 010004 .TTYOUT #'V
33 010014 032702 000004 2$: BIT #4,R2 ;Z-flag?
34 010020 001404 BEQ 3$ ;Br if not
35 010022 .TTYOUT #'Z
36 010032 032702 000010 3$: BIT #10,R2 ;N-flag?
37 010036 001404 BEQ 4$ ;Br if not
38 010040 .TTYOUT #'N
39 ;
40 ; Finished
41 ;
42 010050 005237 0000006 4$: INC D.PCOL ;Advance column number
43 010054 000207 RETURN

```

ODCxxx -- Display instruction operands

```

1 ;-----
2 ; TYP6 -- Branch offset operand
3 ;
4 010056 ODC6:
5 ;
6 ; Tab over to the operand field
7 ;
8 010056 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
9 010062 004737 011734'    CALL     ODCTAB         ;Tab over to operand field
10 ;
11 ; Display the branch target
12 ;
13 010066 010200           MOV      R2,RO          ;Get instruction word
14 010070 042700 177400    BIC     #^C377,RO      ;Leave only branch offset
15 010074 032700 000200    BIT     #200,RO        ;Is offset negative?
16 010100 001402           BEQ     1$              ;Br if not
17 010102 052700 177400    BIS     #177400,RO     ;Sign extend the offset
18 010106 006300          1$: ASL     RO             ;Convert word offset to byte offset
19 010110 060100           ADD     R1,RO          ;Add current PC address
20 010112 004737 011570'    CALL     ODCADR        ;Display the address
21 ;
22 ; Finished
23 ;
24 010116 000207           RETURN
25 ;
26 ;-----
27 ; TYP7 -- Register and general destination.
28 ;
29 010120 ODC7:
30 ;
31 ; Tab over to the operand field
32 ;
33 010120 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
34 010124 004737 011734'    CALL     ODCTAB         ;Tab over to operand field
35 ;
36 ; Display the register
37 ;
38 010130 010200           MOV      R2,RO          ;Get instruction word
39 010132 072027 177772    ASH     #-6,RO         ;Right justify the register value
40 010136 004737 011342'    CALL     ODCREG         ;Display the register name
41 010142           .TTYOUT #COMMA       ;Display ","
42 010152 005237 000000G    INC     D.PCOL         ;Inc column counter
43 ;
44 ; Display the destination address
45 ;
46 010156 010200           MOV      R2,RO          ;Get the destination address descriptor
47 010160 004737 010650'    CALL     ODCGEN         ;Display the address
48 ;
49 ; Finished
50 ;
51 010164 000207           RETURN

```

ODCxxx -- Display instruction operands

```

1 ;-----
2 ; TYPB -- Register destination and general source operand (SS,R)
3 ;
4 010166 ODCB:
5 ;
6 ; Tab over to the operand field
7 ;
8 010166 012700 000010      MOV      #COLDPN,R0      ;Get operand column number
9 010172 004737 011734'    CALL     ODC TAB        ;Tab over to operand field
10 ;
11 ; Display the source item
12 ;
13 010176 010200           MOV      R2,R0          ;Get the source operand code
14 010200 004737 010650'   CALL     ODC GEN        ;Display it
15 010204           .TTYOUT #COMMA      ;Display ","
16 010214 005237 0000000   INC      D.PCOL        ;Advance column counter
17 ;
18 ; Display the register destination
19 ;
20 010220 010200           MOV      R2,R0          ;Get the register number
21 010222 072027 177772   ASH     #-6,R0         ;Right justify
22 010226 004737 011342'   CALL     ODC REG        ;Display the register
23 ;
24 ; Finished
25 ;
26 010232 000207           RETURN

```


ODCxxx -- Display instruction operands

```

1 ;-----
2 ; TYP9 -- Two general operands
3 ;
4 010234 ODC9:
5 ;
6 ; Tab over to the operand field
7 ;
8 010234 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
9 010240 004737 011734'    CALL     ODCTAB          ;Tab over to operand field
10 ;
11 ; Display the source operand
12 ;
13 010244 010200          MOV      R2,RO          ;Get instruction word
14 010246 072027 177772    ASH     #-6,RO          ;Right justify the source item
15 010252 004737 010650'    CALL     ODCGEN         ;Display source item
16 010256                .TTYOUT #COMMA        ;Print a comma
17 ;
18 ; Display the destination operand
19 ;
20 010266 010200          MOV      R2,RO          ;Get destination item code
21 010270 004737 010650'    CALL     ODCGEN         ;Display destination item
22 ;
23 ; Finished
24 ;
25 010274 000207          RETURN

```

ODCxxx --- Display instruction operands

```

1 ;-----
2 ; TYP10 -- SOB instruction
3 ;
4 010276 ODC10:
5 ;
6 ; Tab over to the operand field
7 ;
8 010276 012700 000010      MOV      #COLDPN,R0      ;Get operand column number
9 010302 004737 011734'    CALL     ODCTAB          ;Tab over to operand field
10 ;
11 ; Display the register number
12 ;
13 010306 010200      MOV      R2,R0          ;Get instruction word
14 010310 072027 177772    ASH     #-6,R0          ;Right justify register number
15 010314 004737 011342'    CALL     ODCREG         ;Display the register name
16 010320      .TTYOUT #COMMA      ;Print comma
17 010330 005237 000000G    INC     D.PCOL         ;Advance column number
18 ;
19 ; Display the destination address
20 ;
21 010334 042702 177700      BIC     #^C77,R2       ;Mask out all but branch offset
22 010340 010100      MOV     R1,R0          ;Get the current PC address
23 010342 006302      ASL     R2              ;Get 2* offset
24 010344 160200      SUB     R2,R0          ;Get destination address
25 010346 004737 011570'    CALL     ODCADR         ;Display the address
26 ;
27 ; Finished
28 ;
29 010352 000207      RETURN

```

ODCxxx -- Display instruction operands

```

1 ;-----
2 ; TYP11 -- EMT and TRAP instructions
3 ;
4 010354 ODC11:
5 ;
6 ; Tab over to the operand field
7 ;
8 010354 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
9 010360 004737 011734'    CALL     ODCTAB         ;Tab over to operand field
10 ;
11 ; Display the argument value
12 ;
13 010364 010200           MOV      R2,RO          ;Get instruction word
14 010366 042700 177400    BIC     #^C377,RO      ;Mask out all but operand value
15 010372 004737 005772'    CALL     PRTOCT        ;Print octal value
16 ;
17 ; Finished
18 ;
19 010376 000207           RETURN
20 ;-----
21 ;
22 ; TYP12 -- Single floating-point operand
23 ;
24 010400 ODC12:
25 ;
26 ; Tab over to the operand field
27 ;
28 010400 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
29 010404 004737 011734'    CALL     ODCTAB         ;Tab over to operand field
30 ;
31 ; Display the floating point operand
32 ;
33 010410 010200           MOV      R2,RO          ;Get operand code
34 010412 004737 011542'    CALL     ODCFPU         ;Display general FPU operand
35 ;
36 ; Finished
37 ;
38 010416 000207           RETURN
39 ;-----
40 ;
41 ; TYP13 -- Floating accumulator and floating source operand.
42 ;
43 010420 ODC13:
44 ;
45 ; Tab over to the operand field
46 ;
47 010420 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
48 010424 004737 011734'    CALL     ODCTAB         ;Tab over to operand field
49 ;
50 ; Display the source operand
51 ;
52 010430 010200           MOV      R2,RO          ;Get instruction word
53 010432 004737 011542'    CALL     ODCFPU         ;Display the source operand
54 010436           .TTYOUT #COMMA      ;Display comma
55 010446 005237 000000G    INC     D.PCOL         ;Advance column number
56 ;
57 ; Display the accumulator number

```

```
58 ;  
59 010452 010200 ; MOV R2,R0 ;Get accumulator number  
60 010454 072027 177772 ; ASH #-6,R0 ;Right justify  
61 010460 004737 011470' ; CALL ODCACC ;Display the accumulator number  
62 ;  
63 ; Finished  
64 ;  
65 010464 000207 ; RETURN
```

ODCxxx -- Display instruction operands

```

1 ;-----
2 ; TYP14 -- Floating accumulator and general source operand.
3 ;
4 010466 ODC14:
5 ;
6 ; Tab over to the operand field
7 ;
8 010466 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
9 010472 004737 011734'    CALL     ODCTAB         ;Tab over to operand field
10 ;
11 ; Display the source operand
12 ;
13 010476 010200          MOV      R2,RO          ;Get instruction word
14 010500 004737 010650'  CALL     ODCGEN         ;Display the source operand
15 010504          .TTYOUT #COMMA      ;Display comma
16 010514 005237 000000G   INC      D.PCOL        ;Advance column number
17 ;
18 ; Display the accumulator number
19 ;
20 010520 010200          MOV      R2,RO          ;Get accumulator number
21 010522 072027 177772    ASH     #-6,RO         ;Right justify
22 010526 004737 011470'  CALL     ODCACC         ;Display the accumulator number
23 ;
24 ; Finished
25 ;
26 010532 000207          RETURN

```

ODCxxx -- Display instruction operands

```

1 ;-----
2 ; TYP15 -- Floating accumulator and floating destination operand.
3 ;
4 010534 ODC15:
5 ;
6 ; Tab over to the operand field
7 ;
8 010534 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
9 010540 004737 011734'    CALL     ODC TAB        ;Tab over to operand field
10 ;
11 ; Display the floating accumulator
12 ;
13 010544 010200           MOV      R2,RO          ;Get the accumulator number
14 010546 072027 177772    ASH     #-6,RO         ;Right justify
15 010552 004737 011470'    CALL     ODCACC        ;Display the accumulator
16 010556           .TTYOUT #COMMA      ;Display comma
17 010566 005237 000000G    INC     D.PCOL        ;Advance column number
18 ;
19 ; Display the floating destination
20 ;
21 010572 010200           MOV      R2,RO          ;Get the destination operand code
22 010574 004737 011542'    CALL     ODCFPU        ;Display it
23 ;
24 ; Finished
25 ;
26 010600 000207           RETURN

```

ODCxxx -- Display instruction operands

```

1 ;-----
2 ; TYP16 -- Floating accumulator and general destination operand.
3 ;
4 010602 ODC16:
5 ;
6 ; Tab over to the operand field
7 ;
8 010602 012700 000010      MOV      #COLOPN,RO      ;Get operand column number
9 010606 004737 011734'    CALL     ODCTAB         ;Tab over to operand field
10 ;
11 ; Display the floating accumulator
12 ;
13 010612 010200           MOV      R2,RO          ;Get the accumulator number
14 010614 072027 177772    ASH     #-6,RO         ;Right justify
15 010620 004737 011470'    CALL     ODCACC        ;Display the accumulator
16 010624           .TTYOUT #COMMA      ;Display comma
17 010634 005237 000000G    INC     D.PCOL        ;Advance column number
18 ;
19 ; Display the destination
20 ;
21 010640 010200           MOV      R2,RO          ;Get the destination operand code
22 010642 004737 010650'    CALL     ODCGEN        ;Display it
23 ;
24 ; Finished
25 ;
26 010646 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 010650 010246 ODCGEN: MOV     R2,-(SP)
15 010652 010346      MOV     R3,-(SP)
16         ;
17         ; Get the addressing mode
18         ;
19 010654 010002      MOV     R0,R2
20 010656 072227 177776  ASH     #-2,R2      ;Right justify 2* mode
21 010662 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 010666 010003      MOV     R0,R3      ;Get mode and register value
26 010670 042703 177770  BIC     #^C7,R3     ;Mask out all but register number
27 010674 020327 000007  CMP     R3,#7      ;Is register R7?
28 010700 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 010702 004772 011322'  CALL   @AD7VEC(R2)  ;Call the processing routine
35 010706 000402      BR      9$
36         ;
37         ; Jump to processing routine based on the mode
38         ;
39 010710 004772 011302'  1$:   CALL   @ADRVEC(R2) ;Jump to processing routine
40         ;
41         ; Finished
42         ;
43 010714 012603      9$:   MOV     (SP)+,R3
44 010716 012602      MOV     (SP)+,R2
45 010720 000207      RETURN

```


ODCGEN -- Display general operand value

```

1 ;-----
2 ; Display routines for registers R0 through R6
3 ;
4 ; Mode 0 -- R
5 ;
6 010722 004737 011342' ADRMDO: CALL ODCREG ;Display register
7 010726 000207 RETURN
8 ;
9 ; Mode 1 -- (R)
10 ;
11 010730 010046 ADRMD1: MOV RO, -(SP) ;Save the register number
12 010732 . TTYOUT #'(' ;Open paren
13 010742 012600 MOV (SP)+, RO ;Recover the register number
14 010744 004737 011342' CALL ODCREG ;Display the register number
15 010750 . TTYOUT #'(' ;Close paren
16 010760 000207 RETURN
17 ;
18 ; Mode 2 -- (R)+
19 ;
20 010762 004737 010730' ADRMD2: CALL ADRMD1 ;Print "(r)"
21 010766 . TTYOUT #'+' ;Put in trailing plus sign
22 010776 000207 RETURN
23 ;
24 ; Mode 3 -- @(R)+
25 ;
26 011000 010046 ADRMD3: MOV RO, -(SP) ;Save the register number
27 011002 . TTYOUT #'@ ;Print "@"
28 011012 012600 MOV (SP)+, RO ;Recover the register number
29 011014 004737 010762' CALL ADRMD2 ;Print "(r)+"
30 011020 000207 RETURN
31 ;
32 ; Mode 4 -- -(R)
33 ;
34 011022 010046 ADRMD4: MOV RO, -(SP) ;Save the register number
35 011024 . TTYOUT #'- ;Print "-"
36 011034 012600 MOV (SP)+, RO ;Recover the register number
37 011036 004737 010730' CALL ADRMD1 ;Print "(r)"
38 011042 000207 RETURN
39 ;
40 ; Mode 5 -- @-(R)
41 ;
42 011044 010046 ADRMD5: MOV RO, -(SP) ;Save the register number
43 011046 . TTYOUT #'@ ;Print "@-"
44 011056 . TTYOUT #'- ;
45 011066 012600 MOV (SP)+, RO ;Recover register number
46 011070 004737 010730' CALL ADRMD1 ;Print "(r)"
47 011074 000207 RETURN
48 ;
49 ; Mode 6 -- X(R)
50 ;
51 011076 010046 ADRMD6: MOV RO, -(SP) ;Save the register number
52 011100 106521 MFPD (R1)+ ;Get the vector base address
53 011102 062737 000002 000000G ADD #2, D. ILEN ;Say instruction uses 2 more bytes
54 011110 012600 MOV (SP)+, RO ;Get value to R0
55 011112 004737 011570' CALL ODCADR ;Display the address
56 011116 012600 MOV (SP)+, RO ;Recover the register number
57 011120 004737 010730' CALL ADRMD1 ;Print "(r)"

```

58	011124	000207		RETURN	
59				;	
60				; Mode 7 -- @X(R)	
61				;	
62	011126	010046	ADRMD7:	MOV RO, -(SP)	; Save register number
63	011130			. TTYOUT #'@	; Print "@"
64	011140	012600		MOV (SP)+, RO	; Recover register number
65	011142	004737		CALL ADRMD6	; Print "x(r)"
66	011146	000207		RETURN	

ODCGEN -- Display general operand value

```

1 ;-----
2 ; Display routines for the various modes with R7.
3 ;
4 ; Mode 2 -- #n
5 ;
6 011150 AD7MD2: .TTYOUT #'# ;Print "#"
7 011160 106521 MFPD (R1)+ ;Get the value onto the stack
8 011162 062737 000002 000000G ADD #2,D.ILEN ;Say instruction uses 2 more bytes
9 011170 012600 MOV (SP)+,R0 ;Get the value
10 011172 004737 011672' CALL ODCNEG ;See if value should be printed as neg number
11 011176 103002 BCC 9$ ;Br if it was a negative number
12 011200 004737 005772' CALL PRTOCT ;Print as an octal value
13 011204 000207 9$: RETURN
14 ;
15 ; Mode 3 -- @#address
16 ;
17 011206 AD7MD3: .TTYOUT #'@ ;Print "@#"
18 011216 .TTYOUT #'#
19 011226 106521 MFPD (R1)+ ;Get the address
20 011230 062737 000002 000000G ADD #2,D.ILEN ;Advance the instruction length
21 011236 012600 MOV (SP)+,R0 ;Get the address
22 011240 004737 011570' CALL ODCADR ;Display the address
23 011244 000207 RETURN
24 ;
25 ; Mode 6 -- address
26 ;
27 011246 106521 AD7MD6: MFPD (R1)+ ;Get the address
28 011250 062737 000002 000000G ADD #2,D.ILEN ;Advance the instruction length
29 011256 012600 MOV (SP)+,R0 ;Get the address
30 011260 060100 ADD R1,R0 ;Relocate the address
31 011262 004737 011570' CALL ODCADR ;Display the address
32 011266 000207 RETURN
33 ;
34 ; Mode 7 -- @address
35 ;
36 011270 AD7MD7: .TTYOUT #'@ ;Print "@"
37 011300 000762 BR AD7MD6 ;Go display the address

```

ODCGEN -- Display general operand value

```

1 ;-----
2 ; Jump table used to dispatch to the correct display routine for
3 ; addressing modes using registers R0 through R6.
4 ;
5 011302 010722' ADRVEC: .WORD ADRMD0 ; Mode 0
6 011304 010730' .WORD ADRMD1 ; Mode 1
7 011306 010762' .WORD ADRMD2 ; Mode 2
8 011310 011000' .WORD ADRMD3 ; Mode 3
9 011312 011022' .WORD ADRMD4 ; Mode 4
10 011314 011044' .WORD ADRMD5 ; Mode 5
11 011316 011076' .WORD ADRMD6 ; Mode 6
12 011320 011126' .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 011322 010722' AD7VEC: .WORD ADRMD0 ; Mode 0
18 011324 010730' .WORD ADRMD1 ; Mode 1
19 011326 011150' .WORD AD7MD2 ; Mode 2
20 011330 011206' .WORD AD7MD3 ; Mode 3
21 011332 011022' .WORD ADRMD4 ; Mode 4
22 011334 011044' .WORD ADRMD5 ; Mode 5
23 011336 011246' .WORD AD7MD6 ; Mode 6
24 011340 011270' .WORD AD7MD7 ; Mode 7

```

ODCREG -- Display register name

```

1          .SBTTL  ODCREG -- Display register name
2          ;-----
3          ; ODCREG is called to display the name of a register.
4          ;
5          ; Inputs:
6          ;   R0 = Register number
7          ;
8 011342  010046 ODCREG: MOV     R0,-(SP)
9          ;
10         ; Determine which register this is
11         ;
12 011344  042700 177770         BIC     #^C7,R0         ;Clear all but the register number
13         ;
14         ; See if this is R7 (PC)
15         ;
16 011350  020027 000007         CMP     R0,#7         ;Is register PC?
17 011354  001011         BNE     1$
18 011356         .TTYOUT #'P         ;Display "PC"
19 011366         .TTYOUT #'C
20 011376  000427         BR      9$
21         ;
22         ; See if this is R6 (SP)
23         ;
24 011400  020027 000006 1$:    CMP     R0,#6         ;Is register SP?
25 011404  001011         BNE     2$         ;Br if not
26 011406         .TTYOUT #'S         ;Print "SP"
27 011416         .TTYOUT #'P
28 011426  000413         BR      9$
29         ;
30         ; This is a register in the range R0 to R5.
31         ;
32 011430 2$:    .TTYOUT #'R         ;Print "Rn"
33 011440  011600         MOV     (SP),R0         ;Get register number
34 011442  042700 177770         BIC     #^C7,R0
35 011446  062700 000060         ADD     #'0,R0         ;Convert to ascii digit
36 011452         .TTYOUT         ;Print it
37         ;
38         ; Finished
39         ;
40 011456  062737 000002 000000G 9$:  ADD     #2,D.PCOL         ;Advance print column counter
41 011464  012600         MOV     (SP)+,R0
42 011466  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 011470 010046 ODCACC: MOV      RO, -(SP)
9          ;
10         ; Print ACn
11         ;
12 011472          .TTYOUT #'A
13 011502          .TTYOUT #'C
14 011512 011600  MOV      (SP), RO
15 011514 042700 177774 BIC      #'C3, RO      ;Clear all but ACC #
16 011520 062700 000060 ADD      #'0, RO      ;Convert to ascii char
17 011524          .TTYOUT          ;Print the #
18         ;
19         ; Finished
20         ;
21 011530 062737 000003 000000G ADD      #3, D, PCOL   ;Advance print column number
22 011536 012600          MOV      (SP)+, RO
23 011540 000207          RETURN

```

```
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 011542 010046 ODCFPU: MOV     RO,-(SP)
9          ;
10         ; Determine if it is a floating point accumulator
11         ;
12 011544 032700 000070          BIT     #70,RO          ;Is mode = 0 (accumulator)?
13 011550 001003          BNE     1$              ;Br if not
14 011552 004737 011470'        CALL   ODCACC          ;Display an accumulator name
15 011556 000402          BR      9$
16         ;
17         ; It is not an accumulator
18         ;
19 011560 004737 010650'        1$:   CALL   ODCGEN          ;Display general operand
20         ;
21         ; Finished
22         ;
23 011564 012600          9$:   MOV     (SP)+,RO
24 011566 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 011570 010046 ODCADR: MOV     RO,-(SP)
13 011572 010346         MOV     R3,-(SP)
14         ;
15         ; Determine if address should be shown as a negative number
16         ;
17 011574 004737 011672'         CALL    ODCNEG         ;Should this be shown as a negative number?
18 011600 103031         BCC     9$             ;Br if yes
19         ;
20         ; Determine if address is relative to a relocation base
21         ;
22 011602 032737 000000G 000000G         BIT     #DP$DAA,D.PFMT ;Should address be absolute?
23 011610 001023         BNE     1$             ;Br if yes
24 011612 004737 007262'         CALL    RELADR         ;See if address is within relocation region
25 011616 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 011620 010003         MOV     RO,R3         ;Save the address
31 011622         .TTYOUT #'[         ;Print "[r,address]"
32 011632 010300         MOV     R3,RO         ;Get back address
33 011634 004737 007172'         CALL    SHOADR         ;Display the address
34 011640         .TTYOUT #']'         ;Put in closing bracket
35 011650 062737 000002 000000G         ADD     #2,D.PCOL     ;Advance print column counter
36 011656 000402         BR      9$
37         ;
38         ; Address is not relative to a relocation base
39         ;
40 011660 004737 005772' 1$:     CALL    PRTOCT         ;Display the address
41         ;
42         ; Finished
43         ;
44 011664 012603 9$:     MOV     (SP)+,R3
45 011666 012600         MOV     (SP)+,RO
46 011670 000207         RETURN

```


ODCNEG -- Determine if values should be shown as negative

```

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 011672 ODCNEG:
16         ;
17         ; See if this is a value to be displayed as a negative number
18         ;
19 011672 020027 177776      CMP     RO,#177776      ;Is this a negative value?
20 011676 103414          BLO     9$              ;Br if not
21         ;
22         ; This is a negative value
23         ;
24 011700 010046          MOV     RO,-(SP)
25 011702                .TTYOUT #'-'          ;Print minus sign
26 011712 011600          MOV     (SP),RO      ;Get the value
27 011714 005400          NEG     RO              ;Negate it
28 011716 004737 005772'  CALL    PRTOCT        ;Print it
29 011722 012600          MOV     (SP)+,RO    ;Recover original value
30 011724 000241          CLC
31 011726 000401          BR     10$          ;This is a negative number
32         ;
33         ; Finished
34         ;
35 011730 000261          9$:    SEC
36 011732 000207          10$:   RETURN

```

ODCTAB -- Tab to a specified column

```

1
2
3
4
5
6
7
8
9
10
11
12 011734 010046
13
14
15
16 011736
17 011746 005237 0000000
18 011752 023716 0000000
19 011756 103767
20
21
22
23 011760 012600
24 011762 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

```

INSBAS -- Instruction decoding tables

```

1          .SBTTL  INSBAS -- Instruction decoding tables
2          ;-----
3          ; Table used for decoding a PDP-11 instruction into symbolic form.
4          ; Each instruction is defined by an invocation of the INSTR macro
5          ; which takes 4 arguments. The four arguments are:
6          ;   Argument 1 = 16 bit mask used to mask out the operand portions
7          ;                   of the instruction word.
8          ;   Argument 2 = Value of instruction after mask (arg 1) has been applied.
9          ;   Argument 3 = Operand type code for the instruction.
10         ;   Argument 4 = Symbolic name for the instruction.
11         ;
12         ; Define symbolic names for offsets into an instruction description block:
13         ;
14         ; IB$MSK = 0 ; Mask value
15         ; IB$VAL = 2 ; Instruction value
16         ; IB$TYP = 4 ; Operand type
17         ; IB$LEN = 5 ; Length of instruction mnemonic text string
18         ; IB$NAM = 6 ; Start of instruction mnemonic text string
19         ;
20         .MACRO  INSTR  MASK, VALUE, TYPE, NAME
21         .WORD  MASK
22         .WORD  VALUE
23         .BYTE  TYPE
24         .NCHR  NAMLEN, NAME
25         .BYTE  NAMLEN
26         .ASCII /'NAME/'
27         .EVEN
28         .ENDM  INSTR
29         ;
30         ; Type codes for the instruction operand types:
31         ; TYP1 = No operand.
32         ; TYP2 = Single general operand.
33         ; TYP4 = 4-bit absolute numeric operand.
34         ; TYP5 = Condition code modification instruction.
35         ; TYP6 = Branch offset operand.
36         ; TYP7 = Register source, general operand destination.
37         ; TYP9 = General source and destination operands.
38         ; TYP10 = Register and loop offset (SOB).
39         ; TYP11 = EMT and TRAP instructions.
40         ; TYP12 = Single floating point general operand
41         ; TYP13 = Floating point accumulator and floating source operand
42         ; TYP14 = Floating point accumulator and general source operand
43         ; TYP15 = Floating point accumulator and floating destination operand
44         ; TYP16 = Floating point accumulator and general destination operand
45         ;
46         ; Define the instructions
47         ;
48 011764  INSBAS:
49         ;   Mask      Code      Type      Name
50         ;   -----  -
51 011764  INSTR  000000, 000000, TYP1,  HALT
52 011776  INSTR  000000, 000001, TYP1,  WAIT
53 012010  INSTR  000000, 000002, TYP1,  RTI
54 012022  INSTR  000000, 000003, TYP1,  BPT
55 012034  INSTR  000000, 000004, TYP1,  IOT
56 012046  INSTR  000000, 000005, TYP1,  RESET
57 012062  INSTR  000000, 000006, TYP1,  RTT

```

58 012074	INSTR	000077,	000100,	TYP2,	JMP
59 012106	INSTR	000000,	000207,	TYP1,	RETURN
60 012122	INSTR	000007,	000200,	TYP3,	RTS
61 012134	INSTR	000007,	000230,	TYP4,	SPL
62 012146	INSTR	000000,	000240,	TYP1,	NOP
63 012160	INSTR	000017,	000240,	TYP5,	CL
64 012170	INSTR	000017,	000260,	TYP5,	SE
65 012200	INSTR	000077,	000300,	TYP2,	SWAB
66 012212	INSTR	000377,	000400,	TYP6,	BR
67 012222	INSTR	000377,	001000,	TYP6,	BNE
68 012234	INSTR	000377,	001400,	TYP6,	BEQ
69 012246	INSTR	000377,	002000,	TYP6,	BGE
70 012260	INSTR	000377,	002400,	TYP6,	BLT
71 012272	INSTR	000377,	003000,	TYP6,	BGT
72 012304	INSTR	000377,	003400,	TYP6,	BLE
73 012316	INSTR	000077,	004700,	TYP2,	CALL
74 012330	INSTR	000777,	004000,	TYP7,	JSR
75 012342	INSTR	000077,	005000,	TYP2,	CLR
76 012354	INSTR	000077,	005100,	TYP2,	COM
77 012366	INSTR	000077,	005200,	TYP2,	INC
78 012400	INSTR	000077,	005300,	TYP2,	DEC
79 012412	INSTR	000077,	005400,	TYP2,	NEG
80 012424	INSTR	000077,	005500,	TYP2,	ADC
81 012436	INSTR	000077,	005600,	TYP2,	SBC
82 012450	INSTR	000077,	005700,	TYP2,	TST
83 012462	INSTR	000077,	006000,	TYP2,	ROR
84 012474	INSTR	000077,	006100,	TYP2,	ROL
85 012506	INSTR	000077,	006200,	TYP2,	ASR
86 012520	INSTR	000077,	006300,	TYP2,	ASL
87 012532	INSTR	000077,	006400,	TYP3,	MARK
88 012544	INSTR	000077,	006500,	TYP2,	MFP I
89 012556	INSTR	000077,	006600,	TYP2,	MTP I
90 012570	INSTR	000077,	006700,	TYP2,	SXT
91 012602	INSTR	007777,	010000,	TYP9,	MOV
92 012614	INSTR	007777,	020000,	TYP9,	CMP
93 012626	INSTR	007777,	030000,	TYP9,	BIT
94 012640	INSTR	007777,	040000,	TYP9,	BIC
95 012652	INSTR	007777,	050000,	TYP9,	BIS
96 012664	INSTR	007777,	060000,	TYP9,	ADD
97 012676	INSTR	000777,	070000,	TYP3,	MUL
98 012710	INSTR	000777,	071000,	TYP3,	DIV
99 012722	INSTR	000777,	072000,	TYP3,	ASH
100 012734	INSTR	000777,	073000,	TYP3,	ASHC
101 012746	INSTR	000777,	074000,	TYP7,	XOR
102 012760	INSTR	000007,	075000,	TYP1,	FADD
103 012772	INSTR	000007,	075010,	TYP1,	FSUB
104 013004	INSTR	000007,	075020,	TYP1,	FMUL
105 013016	INSTR	000007,	075030,	TYP1,	FDIV
106 013030	INSTR	000777,	077000,	TYP10,	SOB
107 013042	INSTR	000377,	100000,	TYP6,	BPL
108 013054	INSTR	000377,	100400,	TYP6,	BMI
109 013066	INSTR	000377,	101000,	TYP6,	BHI
110 013100	INSTR	000377,	101400,	TYP6,	BLOS
111 013112	INSTR	000377,	102000,	TYP6,	BVC
112 013124	INSTR	000377,	102400,	TYP6,	BVS
113 013136	INSTR	000377,	103000,	TYP6,	BCC
114 013150	INSTR	000377,	103400,	TYP6,	BCS

115	013162	INSTR	000377,	104000,	TYP11,	EMT
116	013174	INSTR	000377,	104400,	TYP11,	TRAP
117	013206	INSTR	000077,	105000,	TYP2,	CLRB
118	013220	INSTR	000077,	105100,	TYP2,	COMB
119	013232	INSTR	000077,	105200,	TYP2,	INCB
120	013244	INSTR	000077,	105300,	TYP2,	DECB
121	013256	INSTR	000077,	105400,	TYP2,	NEGB
122	013270	INSTR	000077,	105500,	TYP2,	ADCB
123	013302	INSTR	000077,	105600,	TYP2,	SBCB
124	013314	INSTR	000077,	105700,	TYP2,	TSTB
125	013326	INSTR	000077,	106000,	TYP2,	RORB
126	013340	INSTR	000077,	106100,	TYP2,	ROLB
127	013352	INSTR	000077,	106200,	TYP2,	ASRB
128	013364	INSTR	000077,	106300,	TYP2,	ASLB
129	013376	INSTR	000077,	106500,	TYP2,	MFPD
130	013410	INSTR	000077,	106600,	TYP2,	MTPD
131	013422	INSTR	007777,	110000,	TYP9,	MOVB
132	013434	INSTR	007777,	120000,	TYP9,	CMPB
133	013446	INSTR	007777,	130000,	TYP9,	BITB
134	013460	INSTR	007777,	140000,	TYP9,	BICB
135	013472	INSTR	007777,	150000,	TYP9,	BISB
136	013504	INSTR	007777,	160000,	TYP9,	SUB
137	013516	INSTR	000000,	170000,	TYP1,	CFCC
138	013530	INSTR	000000,	170001,	TYP1,	SETF
139	013542	INSTR	000000,	170002,	TYP1,	SETI
140	013554	INSTR	000000,	170011,	TYP1,	SETD
141	013566	INSTR	000000,	170012,	TYP1,	SETL
142	013600	INSTR	000077,	170100,	TYP2,	LDFPS
143	013614	INSTR	000077,	170200,	TYP2,	STFPS
144	013630	INSTR	000077,	170300,	TYP2,	STST
145	013642	INSTR	000077,	170400,	TYP12,	CLRF
146	013654	INSTR	000077,	170500,	TYP12,	TSTF
147	013666	INSTR	000077,	170600,	TYP12,	ABSF
148	013700	INSTR	000077,	170700,	TYP12,	NEGF
149	013712	INSTR	000377,	171000,	TYP13,	MULF
150	013724	INSTR	000377,	171400,	TYP13,	MODF
151	013736	INSTR	000377,	172000,	TYP13,	ADDF
152	013750	INSTR	000377,	172400,	TYP13,	LDF
153	013762	INSTR	000377,	173000,	TYP13,	SUBF
154	013774	INSTR	000377,	173400,	TYP13,	CMPF
155	014006	INSTR	000377,	174000,	TYP15,	STF
156	014020	INSTR	000377,	174400,	TYP13,	DIVF
157	014032	INSTR	000377,	175000,	TYP16,	STEXP
158	014046	INSTR	000377,	175400,	TYP16,	STCFI
159	014062	INSTR	000377,	176000,	TYP16,	STCFD
160	014076	INSTR	000377,	176400,	TYP14,	LDEXP
161	014112	INSTR	000377,	177000,	TYP14,	LDCIF
162	014126	INSTR	000377,	177400,	TYP13,	LDCDF
163	014142	INSTR	000077,	106400,	TYP2,	MTPS
164	014154	INSTR	000077,	106700,	TYP2,	MFPS
165	014166	INSTR	177777,	000000,	TYP1,	???
166	014200	INSEND:				

167 ;
 168 ; Address vector used to jump to correct routine to decode and display
 169 ; the operands associated with an instruction.
 170 ;
 171 . MACRO TYPDEF TYPNAM,RTN

```
172          TYPNAM =          .-TYPVEC
173          .WORD   RTN
174          .ENDM   TYPDEF
175          ;
176          ; Define the types
177          ;
178 014200     TYPVEC:
179 014200     TYPDEF TYP1,ODC1      ;No operand
180 014202     TYPDEF TYP2,ODC2      ;Single general operand.
181 014204     TYPDEF TYP3,ODC3      ;Single register operand (RTS)
182 014206     TYPDEF TYP4,ODC4      ;4-bit absolute numeric operand.
183 014210     TYPDEF TYP5,ODC5      ;Condition code modification instruction.
184 014212     TYPDEF TYP6,ODC6      ;Branch offset operand.
185 014214     TYPDEF TYP7,ODC7      ;Register source, general operand destination.
186 014216     TYPDEF TYP8,ODC8      ;Register destination, general source operand
187 014220     TYPDEF TYP9,ODC9      ;General source and destination operands.
188 014222     TYPDEF TYP10,ODC10     ;Register and loop offset (SOB).
189 014224     TYPDEF TYP11,ODC11     ;EMT and TRAP instructions.
190 014226     TYPDEF TYP12,ODC12     ;One floating point operand.
191 014230     TYPDEF TYP13,ODC13     ;Floating accum and floating source operand.
192 014232     TYPDEF TYP14,ODC14     ;Floating accum and general source operand.
193 014234     TYPDEF TYP15,ODC15     ;Floating accumulator and floating destination
194 014236     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: 9616 Words (38 Pages)
Size of core pool: 17920 Words (70 Pages)
Operating system: RT-11

Elapsed time: 00:01:39.45
DK: TSDBUG, LP: TSDBUG=DK: TSDBUG, MAC/C/N: SYM

D. R7	1-27 26-73	3-29 27-14	4-16* 27-61	5-25* 27-66	5-37 29-62	7-8* 29-55	7-34* 41-23	7-70* 42-23	8-62 42-25	16-7* 42-28	17-26* 17-26*	17-37
D. RLBS	1-25	3-34	13-23	14-23*	14-37							
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 20-20	9-13* 21-6	9-22* 22-15	11-9 36-13	13-11	14-29	16-5	16-16	17-13	17-15*	18-57*	20-6
D. V2FL	1-26	9-15*	9-33*	14-13	15-5	16-27	19-15					
D. VAL1	1-26 20-8	9-14* 20-22	9-21* 22-17	13-10 36-16	14-23	14-31	15-19	16-7	16-18	17-21	19-6	19-9
D. VAL2	1-26	9-16*	9-32*	14-18	15-7	16-29	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 23-25	5-39 27-7	9-5 27-28	12-9 27-41	16-8 27-56	16-19 27-68	17-65 35-6	20-32 35-7	21-14 43-9#	21-19	22-22	22-27
DP%DAA	1-35	63-22										
DP%LAA	1-35	40-16										
EM%IIIV	1-88#	29-74										
EM%IRB	1-89#	29-51										
EM%IRC	1-85#	18-34										
EM%IVC	1-86#	9-43										
EM%IVL	1-87#	14-21	14-34	15-11								
EM%NTL	1-90#	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 31-8#	9-37	18-13	18-24	18-39	18-54	29-19	29-53	29-68	30-17	30-30	30-71
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#											
INSLN	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	1-70#	1-91	2-19	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 14-42 27-16	5-36 14-45 27-19	9-9 14-53 27-34	10-14 18-5 27-46	10-28 18-8 27-49	10-34 20-29 27-57	12-19 20-34 35-20#	12-25 21-16 35-24	13-17 21-21 45-24	13-20 22-24	13-31 22-29	14-28 27-13
LSW9	1-33	6-16*	6-17*									
NAMLEN	66-51	66-51#	66-52	66-52#	66-53	66-53#	66-54	66-54#	66-55	66-55#	66-56	66-56#

	66-57	66-57#	66-58	66-58#	66-59	66-59#	66-60	66-60#	66-61	66-61#	66-62	66-62#
	66-63	66-63#	66-64	66-64#	66-65	66-65#	66-66	66-66#	66-67	66-67#	66-68	66-68#
	66-69	66-69#	66-70	66-70#	66-71	66-71#	66-72	66-72#	66-73	66-73#	66-74	66-74#
	66-75	66-75#	66-76	66-76#	66-77	66-77#	66-78	66-78#	66-79	66-79#	66-80	66-80#
	66-81	66-81#	66-82	66-82#	66-83	66-83#	66-84	66-84#	66-85	66-85#	66-86	66-86#
	66-87	66-87#	66-88	66-88#	66-89	66-89#	66-90	66-90#	66-91	66-91#	66-92	66-92#
	66-93	66-93#	66-94	66-94#	66-95	66-95#	66-96	66-96#	66-97	66-97#	66-98	66-98#
	66-99	66-99#	66-100	66-100#	66-101	66-101#	66-102	66-102#	66-103	66-103#	66-104	66-104#
	66-105	66-105#	66-106	66-106#	66-107	66-107#	66-108	66-108#	66-109	66-109#	66-110	66-110#
	66-111	66-111#	66-112	66-112#	66-113	66-113#	66-114	66-114#	66-115	66-115#	66-116	66-116#
	66-117	66-117#	66-118	66-118#	66-119	66-119#	66-120	66-120#	66-121	66-121#	66-122	66-122#
	66-123	66-123#	66-124	66-124#	66-125	66-125#	66-126	66-126#	66-127	66-127#	66-128	66-128#
	66-129	66-129#	66-130	66-130#	66-131	66-131#	66-132	66-132#	66-133	66-133#	66-134	66-134#
	66-135	66-135#	66-136	66-136#	66-137	66-137#	66-138	66-138#	66-139	66-139#	66-140	66-140#
	66-141	66-141#	66-142	66-142#	66-143	66-143#	66-144	66-144#	66-145	66-145#	66-146	66-146#
	66-147	66-147#	66-148	66-148#	66-149	66-149#	66-150	66-150#	66-151	66-151#	66-152	66-152#
	66-153	66-153#	66-154	66-154#	66-155	66-155#	66-156	66-156#	66-157	66-157#	66-158	66-158#
	66-159	66-159#	66-160	66-160#	66-161	66-161#	66-162	66-162#	66-163	66-163#	66-164	66-164#
	66-165	66-165#										
NEWCMD	5-55	7-93	9-9#	20-11	35-10							
NEWLIN	9-5#	11-14	14-24	15-26	16-32	18-62	19-22	20-36	21-23	22-14		
NUMCMD	2-23#	9-38										
NUMINT	3-17#	29-69										
ODC1	46-5#	66-179										
ODC10	51-4#	66-188										
ODC11	52-4#	66-189										
ODC12	52-24#	66-190										
ODC13	52-43#	66-191										
ODC14	53-4#	66-192										
ODC15	54-4#	66-193										
ODC16	55-4#	66-194										
ODC2	46-10#	66-180										
ODC3	46-29#	66-181										
ODC4	47-4#	66-182										
ODC5	47-23#	66-183										
ODC6	48-4#	66-184										
ODC7	48-29#	66-185										
ODC8	49-4#	66-186										
ODC9	50-4#	66-187										
ODCACC	52-61	53-22	54-15	55-15	61-8#	62-14						
ODCADR	48-20	51-25	57-55	58-22	58-31	63-12#						
ODCFPU	52-34	52-53	54-22	62-8#								
ODCGEN	46-20	48-47	49-14	50-15	50-21	53-14	55-22	56-14#	62-19			
ODCNEG	58-10	63-17	64-15#									
ODCREG	46-39	48-40	49-22	51-15	57-6	57-14	60-8#					
ODCTAB	11-31	46-15	46-34	47-9	48-9	48-34	49-9	50-9	51-9	52-9	52-29	52-48
	53-9	54-9	55-9	65-12#								
OFFEMT	1-65#	44-13										
ONEMT	1-64#	44-5										
OVRHC	1-32	43-23	43-30	43-32								
PRTBYT	10-27	33-31#										
PRTOCT	13-19	13-24	14-41	14-44	27-48	32-8#	39-62	40-26	41-26	41-33	47-14	52-15
	58-12	63-40	64-28									
PRTR50	18-7	34-8#										
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	5-00*	23-21*									
PUTCHR	1-32	43-23	43-30	43-32								
R50CHR	1-97#	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	1-74#	65-16										
STRVAL	18-49	20-9	20-23	22-18	37-11#							
TAB	1-73#	2-22										
TM#GRT	1-91#	23-25										
TRCTRP	1-79#	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	1-44#	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				
OCALL	1-39#	43-23	43-30	43-32								
PRINT	1-56#	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	1-49#	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							