```
    4-    1       DBGENT -- Enter debugger to start program
    5-    1       DBGTRP -- Entry from trap
    6-    1       DBGBRK -- Enter debugger by typing ctrl-D
    7-    1       BRKENT -- Enter debugger from breakpoint
    8-    1       EXIT   -- Exit from debugger to user
    9-    1       NEWCMD -- Get new command
   10-    1       -- Command processing routines --
   10-    2       "/"  -- Examine word
   10-   17       "\"  -- Examine byte
   11-    1       "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       "]"  -- Close location and advance to instruction
   22-    1       "^"  -- Close current location and go to previous
   23-    1       -- Subroutines --
   23-    2       DBGINI -- Initialize the debugger
   24-    1       BRKRST -- Reset breakpoint instructions
   25-    1       BRKSET -- Set breakpoints
   26-    1       BRKCHK -- Determine type of breakpoint
   27-    1       SHOBRK -- Display breakpoint information
   28-    1       FLGBRK -- Set flag if any breakpoints are active
   29-    1       ACRVAL -- Accrue a general value
   30-    1       ACRNUM -- Accrue a number
   31-    1       GETCHR -- Get next character from terminal
   31-   34       PSHCHR -- Push a character
   32-    1       PRTOCT -- Print octal value
   33-    1       PRTWRD -- Print octal word value
   33-   24       PRTBYT -- Print octal byte value
   34-    1       PRTR50 -- Print a RAD50 value
   35-    1       ERRPRT -- Print an error message
   35-   12       LSTTXT -- Print text string
   36-    1       SETLOC -- Set address of open cell
   37-    1       STRVAL -- Store value into open cell
   38-    1       GETVAL -- Get value from current cell
   39-    1       SHOLOC -- Display current address
   40-    1       LSTADR -- Display address in user's program
   41-    1       SHOADR -- Display address in user's program
   42-    1       RELADR -- Determine if address is in relocation region
   43-    1       DOPRT  -- Print a text string
   44-    1       DBGON  -- Place terminal in debug mode
   44-    9       DBGOFF -- Turn off terminal debug mode
   44-   17       CHKADR -- Determine if address is valid
   45-    1       DECODE -- Decode instruction into symbolic form
   46-    1       ODCxxx -- Display instruction operands
   56-    1       ODCGEN -- Display general operand value
   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

```
                                        .TITLE  TSDBUG -- TSX-Plus debugging module
                                        .ENABL  LC
                                        .DSABL  GBL
                                        .ENABL  AMA
000000                                  .CSECT  TSDBUG
000000  014527                  TSDBUG: .RAD50  /DBG/                   ;System overlay id
                                ;-------------------------------------------------------------
                                ;  This module implements the TSX-Plus program debugging facility.
                                ;
                                ;  Copyright (c) 1984.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee  USA
                                ;  All rights reserved.
                                ;
                                ;  Macro calls
                                ;
                                        .MCALL  .TTYIN,.TTYOUT
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  DBGENT,BRKENT,DBGBRK,DBGTRP
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  D.BYTM,D.BKAD,D.RLBS,D.SVCH,D.NMBF,D.LOC
                                        .GLOBL  D.VAL1,D.V1FL,D.VAL2,D.V2FL,D.LOCM
                                        .GLOBL  D.R0,D.R1,D.R2,D.R3,D.R4,D.R5,D.R6,D.R7
                                        .GLOBL  D.FLAG,D$SSTP,D$DBRK,D$IBRK,D$SBRK,D.BKNM
                                        .GLOBL  D.BKSV,D$DMON,D.DADR,D.DOLD,D.DTRG
                                        .GLOBL  D$DVAL,D$IPND,D$BKST,D.PS,UMODE,UPMODE
                                        .GLOBL  D.SPSV,D.NMBE,D.START,D.END,D$FBRK
                                        .GLOBL  PUTCHR,CORUSR,OVRHC,UPMODE,PSW,D.PCNT
                                        .GLOBL  LSW9,$DBGBK,D$RUN,D.LVAL,D.MASK,D$CKBK
                                        .GLOBL  D$INIT,$DEBUG,D.PCOL,D.ILEN,D$TSTP
                                        .GLOBL  D.PFMT,DP$DAA,DP$LAA,D.CBRK
                                ;
                                ;  Macro definitions
                                ;
                                        .MACRO  OCALL   ENTADD
                                        CALL    OVRHC
                                        .WORD   ENTADD
                                        .ENDM   OCALL
                                ;
                                        .MACRO  ERR     ERRMSG
                                        MOV     ERRMSG,R1
                                        JMP     ERRPRT
                                        .ENDM   ERR
                                ;
                                        .MACRO  TPRINT  STRING
                                        .IF     NB STRING
                                        MOV     STRING,R0
                                        .ENDC
                                        CALL    DOPRT
                                        .ENDM   TPRINT
                                ;
                                        .MACRO  PRINT   STRING
                                        JSR     R2,LSTTXT
```

```
                                        .ASCIZ  "'STRING'"
                                        .EVEN
                                        .ENDM   PRINT
                                ;
                                ;  EMT argument blocks to control terminal debug mode
                                ;
000002      001     111         ONEMT:  .BYTE   1,111           ;Turn on debug mode
000004      000     111         OFFEMT: .BYTE   0,111           ;Turn off debug mode
                                ;
                                ;  Ascii character values
                                ;
            000015              CR      =       15              ;Carriage-return
            000012              LF      =       12              ;Line-feed
            000007              BELL    =       07              ;Bell
            000010              BKSPAC  =       10              ;Backspace
            000011              TAB     =       11              ;Tab
            000040              SPACE   =       40              ;Space
            000054              COMMA   =       54              ;Comma
                                ;
                                ;  Assembly parameters
                                ;
            000020              TRCTRP  =       20              ;Trace trap flag in PSW
            000010              COLOPN  =       8.              ;Column number where operand is to go
                                ;
                                ;  Error messages
                                ;
                                        .NLIST  BEX
000006      111     156     166 EM$IRC: .ASCIZ   /Invalid RAD50 character/
000036      111     156     166 EM$IVC: .ASCIZ   /Invalid command/
000056      111     156     166 EM$IVL: .ASCIZ   /Invalid value/
000074      111     156     166 EM$IIV: .ASCIZ   /Invalid register/
000115      111     156     166 EM$IRB: .ASCIZ   /Invalid relocation register/
000151      116     165     155 EM$NTL: .ASCIZ   /Number too long/
000171      015     012     124 TM$GRT: .ASCIZ   <CR><LF>/TSX-Plus debugger/
000215      000                 CRLF:   .BYTE   0
000216      015     200         CRCHAR: .BYTE   CR,200          ;Print CR only
                                ;
                                ;  RAD50 character table
                                ;
000220      040     101     102 R50CHR: .ASCII  / ABCDEFGHIJKLMNOPQRSTUVWXYZ$.*0123456789/
                                        .EVEN
                                        .LIST   BEX
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Table of command characters
                                ;
000270      057                 CMDTBL: .BYTE   '/              ; /
000271      134                         .BYTE   '\              ; \
000272      100                         .BYTE   '@              ; @
000273      137                         .BYTE   '_              ; _
000274      041                         .BYTE   '!              ; !
000275      133                         .BYTE   '[              ; [
000276      135                         .BYTE   ']              ; ]
000277      102                         .BYTE   'B              ; B
000300      107                         .BYTE   'G              ; G
000301      115                         .BYTE   'M              ; M
000302      120                         .BYTE   'P              ; P
000303      122                         .BYTE   'R              ; R
000304      123                         .BYTE   'S              ; S
000305      130                         .BYTE   'X              ; X
000306      015                         .BYTE   CR              ; Carriage-return
000307      012                         .BYTE   LF              ; Line-feed
000310      136                         .BYTE   '^              ; Up arrow
000311      010                         .BYTE   BKSPAC          ; Backspace
000312      011                         .BYTE   TAB             ; Tab
            000023              NUMCMD  =       .-CMDTBL
                                        .EVEN
                                ;
                                ;  Parallel table of command processing routines
                                ;
000314  001650'                 CMDVEC: .WORD   CMDSLH          ; /
000316  001726'                         .WORD   CMDBKS          ; \
000320  002134'                         .WORD   CMDAT           ; @
000322  002230'                         .WORD   CMDUS           ; _
000324  002250'                         .WORD   CMDEXP          ; !
000326  002034'                         .WORD   CMDDCD          ; [
000330  003672'                         .WORD   CMDRSB          ; ]
000332  002554'                         .WORD   CMDB            ; B
000334  002656'                         .WORD   CMDG            ; G
000336  003424'                         .WORD   CMDM            ; M
000340  002712'                         .WORD   CMDP            ; P
000342  002352'                         .WORD   CMDR            ; R
000344  002742'                         .WORD   CMDS            ; S
000346  003166'                         .WORD   CMDX            ; X
000350  003504'                         .WORD   CMDCR           ; Carriage-return
000352  003536'                         .WORD   CMDLF           ; Line-feed
000354  004034'                         .WORD   CMDUP           ; Up arrow
000356  004022'                         .WORD   CMDBSP          ; Backspace
000360  002776'                         .WORD   CMDTAB          ; Tab
```

```
                                ;-----------------------------------------------------------------------
                                ;  Table of internal register names
                                ;
000362      060         INTCHR: .BYTE   '0              ; $0
000363      061                 .BYTE   '1              ; $1
000364      062                 .BYTE   '2              ; $2
000365      063                 .BYTE   '3              ; $3
000366      064                 .BYTE   '4              ; $4
000367      065                 .BYTE   '5              ; $5
000370      066                 .BYTE   '6              ; $6 (SP)
000371      067                 .BYTE   '7              ; $7 (PC)
000372      123                 .BYTE   'S              ; $S (PS)
000373      115                 .BYTE   'M              ; $M (Mask word)
000374      106                 .BYTE   'F              ; $F (Format register)
000375      102                 .BYTE   'B              ; $B (Breakpoint locations)
000376      122                 .BYTE   'R              ; $R (Relocation base addresses)
            000015      NUMINT  =       .-INTCHR
                                .EVEN
                                ;
                                ;  Parallel table of addresses of internal register cells
                                ;
000400  0000000         INTADR: .WORD   D.R0            ; $0
000402  0000000                 .WORD   D.R1            ; $1
000404  0000000                 .WORD   D.R2            ; $2
000406  0000000                 .WORD   D.R3            ; $3
000410  0000000                 .WORD   D.R4            ; $4
000412  0000000                 .WORD   D.R5            ; $5
000414  0000000                 .WORD   D.R6            ; $6 (SP)
000416  0000000                 .WORD   D.R7            ; $7 (PC)
000420  0000000                 .WORD   D.PS            ; $S (PS)
000422  0000000                 .WORD   D.MASK          ; $M (Mask word)
000424  0000000                 .WORD   D.PFMT          ; $F (Format register)
000426  0000000                 .WORD   D.BKAD          ; $B (Breakpoint locations)
000430  0000000                 .WORD   D.RLBS          ; $R (Relocation base addresses)
000432                  INTEND:                         ; End of table of addresses
```

```
                                        .SBTTL  DBGENT -- Enter debugger to start program
                                ;-----------------------------------------------------------------------
                                ;  DBGENT is jumped to when a program is started under the debugger.
                                ;  The following information is on the stack:
                                ;    0(SP) = PC of start point.
                                ;    2(SP) = PS to start with.
                                ;
000432                          DBGENT:
                                ;
                                ;  Initialize the debugger
                                ;
000432  004737  004170'                 CALL    DBGINI          ;Initialize the debugger
                                ;
                                ;  Set starting address and initial PSW
                                ;
000436  012637  0000000                 MOV     (SP)+,D.R7      ;Set starting address
000442  012637  0000000                 MOV     (SP)+,D.PS      ;Set starting PSW
                                ;
                                ;  Enter debugger
                                ;
000446  000137  001052'                 JMP     DBGRUN          ;Enter debugger
```

```
                                        .SBTTL  DBGTRP -- Entry from trap
                                ;-----------------------------------------------------------------------
                                ;  DBGTRP is jumped to if a trap occurs in a program running under the
                                ;  debugger.
                                ;  On entry, the following values are set up:
                                ;    0(SP) = Saved R5
                                ;    2(SP) = Saved R4
                                ;    4(SP) = Trap PC
                                ;    6(SP) = Trap PS
                                ;    R5    = Trap code (1==>Trap 4, 2==>Trap 10)
                                ;    R4    = Non-zero ==> Trap within debugger
                                ;
000452                          DBGTRP:
                                ;
                                ;  See if the trap occurred in the debugger or in the user's program
                                ;
000452  005704                          TST     R4              ;Trap in debugger or user's program?
000454  001077                          BNE     10$             ;Br if trap occurred in the debugger
                                ;
                                ;  Trap occurred in the user's program.
                                ;  Save entry information.
                                ;
000456  012637  0000000                 MOV     (SP)+,D.R5
000462  012637  0000000                 MOV     (SP)+,D.R4
000466  012637  0000000                 MOV     (SP)+,D.R7      ;Trap PC
000472  012637  0000000                 MOV     (SP)+,D.PS      ;Trap PSW
000476  010037  0000000                 MOV     R0,D.R0
                                ;
                                ;  Print trap entry message
                                ;
000502                                  TPRINT  #CRLF           ;Go to a new line
000512  020527  000001                  CMP     R5,#1           ;Trap to 4 or 10?
000516  001017                          BNE     1$              ;Br if trap to 10
000520                                  PRINT   <Trap to 4 at location >
000554  000416                          BR      2$
000556                          1$:     PRINT   <Trap to 10 at location >
000612  013700  0000000         2$:     MOV     D.R7,R0         ;Get address where trap occurred
000616  004737  007146'                 CALL    LSTADR          ;Display the address
000622                          4$:     TPRINT  #CRLF           ;Go to new line
                                ;
                                ;  Enter the debugger
                                ;
000632  042737  000000C 0000000         BIC     #<D$DBRK!D$IBRK!D$SBRK>,D.FLAG ;Say no breakpoint occurred
000640  013704  0000000                 MOV     D.R4,R4         ;Restore R4 and R5
000644  013705  0000000                 MOV     D.R5,R5
000650  000137  001052'                 JMP     DBGRUN          ;Enter the debugger
                                ;
                                ;  A trap occurred within the debugger.
                                ;  Restore context to condition at the time of the trap and then
                                ;  reenter the debugger.
                                ;
000654  013706  0000000         10$:    MOV     D.SPSV,SP       ;Restore the stack pointer
000660  052737  0000000 0000000         BIS     #UPMODE,@#PSW   ;Make sure previous mode = user
000666  005037  0000000                 CLR     D.LOC           ;No currently open cell
000672  000137  001506'                 JMP     NEWCMD          ;Reenter the debugger
```

```
                                        .SBTTL  DBGBRK -- Enter debugger by typing ctrl-D
                                ;+-----------------------------------------------------------------------
                                ;  DBGBRK is entered when the user requests entry to the debugger by typing
                                ;  ctrl-D.
                                ;
                                ;  On entry, the stack is set up as follows:
                                ;    0(SP) = PC of next instruction.
                                ;    2(SP) = PS of next instruction.
                                ;
000676                          DBGBRK:
                                ;
                                ;  Reset job control flag that was used to force the breakpoint
                                ;
000676  010146                          MOV     R1,-(SP)
000700  113701  0000000                 MOVB    CORUSR,R1       ;Get job index number
000704  042761  0000000 0000000         BIC     #$DBGBK,LSW9(R1);Clear flag saying to force debugger break
000712  052761  0000000 0000000         BIS     #$DEBUG,LSW9(R1);Say program is now running with debugger
000720  012601                          MOV     (SP)+,R1
                                ;
                                ;  If we have not initialized the debugger, do the debugger startup
                                ;
000722  032737  0000000 0000000         BIT     #D$INIT,D.FLAG  ;Has initialization been done?
000730  001002                          BNE     1$              ;Br if yes
000732  004737  004170'                 CALL    DBGINI          ;Initialize the debugger
                                ;
                                ;  Enter debugger as if we hit a breakpoint
                                ;
000736  052737  0000000 0000000 1$:     BIS     #D$FBRK,D.FLAG  ;Set flag saying we had a forced break
000744  000137  000750'                 JMP     BRKENT          ;Now enter debugger as if we had a breakpoint
```

```
                                        .SBTTL  BRKENT -- Enter debugger from breakpoint
                                ;-----------------------------------------------------------------------
                                ;  BRKENT is jumped to when a breakpoint trap occurs.
                                ;  The following information is on the stack:
                                ;   0(SP) = PC of breakpoint trap.
                                ;   2(SP) = PS of breakpoint trap.
                                ;
000750  011637  0000000         BRKENT: MOV     (SP),D.R7       ;Save breakpoint PC
000754  010037  0000000                 MOV     R0,D.R0         ;Save user's R0
                                ;
                                ;  Determine if we should enter the debugger
                                ;
000760  004737  004420'                 CALL    BRKCHK          ;See if breakpoint causes program stop
000764  032737  000000C 0000000         BIT     #<D$DBRK!D$IBRK!D$SBRK!D$FBRK>,D.FLAG ;Breakpoint?
000772  001023                         BNE     5$              ;Br if this break will enter debugger
                                ;
                                ;  This breakpoint will not enter the debugger.
                                ;  If we just executed an instruction that was under a breakpoint,
                                ;  set breakpoints and then continue program execution.
                                ;
000774  042737  0000000 0000000         BIC     #D$IPND,D.FLAG  ;Say pending instruction has been executed
001002  032737  000000C 0000000         BIT     #<D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Are we single stepping
001010  001006                         BNE     3$              ;Br if yes
001012  004737  004340'                 CALL    BRKSET          ;Set breakpoints
001016  042766  000020  000002         BIC     #TRCTRP,2(SP)   ;Reset trace trap flag in PSW
001024  000403                         BR      2$
001026  052766  000020  000002  3$:    BIS     #TRCTRP,2(SP)   ;Set trap flag in PSW on stack
001034  013700  0000000         2$:    MOV     D.R0,R0         ;Recover R0
001040  000006                         RTT                     ;Return from breakpoint
                                ;
                                ;  This breakpoint will enter the debugger.
                                ;  Save information and enter the debugger
                                ;
001042  012637  0000000         5$:    MOV     (SP)+,D.R7      ;Save trap PC
001046  012637  0000000                 MOV     (SP)+,D.PS      ;Save trap PS
                                ;
                                ;  Save user's registers
                                ;
001052  052737  0000000 0000000 DBGRUN: BIS     #D$RUN,D.FLAG   ;Set flag saying debugger is running
001060  010137  0000000                 MOV     R1,D.R1
001064  010237  0000000                 MOV     R2,D.R2
001070  010337  0000000                 MOV     R3,D.R3
001074  010437  0000000                 MOV     R4,D.R4
001100  010537  0000000                 MOV     R5,D.R5
001104  106506                         MFPD    SP
001106  012637  0000000                 MOV     (SP)+,D.R6
                                ;
                                ;  Save initial stack pointer in case we need to restore it if an
                                ;  error occurs.
                                ;
001112  010637  0000000                 MOV     SP,D.SPSV       ;Save initial stack pointer
                                ;
                                ;  Clear the trace-trap flag in the user's TRAP instruction PSW.
                                ;  We may have set the trace-trap flag in this PSW if we were single
                                ;  stepping the program.
                                ;
001116  106537  000036                  MFPD    @#36            ;Get user's TRAP PSW
```

```
      58 001122  042716  000020                    BIC     #TRCTRP,(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  000000G 000000G          BIC     #D$IPND,D.FLAG  ;Clear instruction-pending flag
      65 001140  032737  000000C 000000G          BIT     #<D$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  000000G 000000G          BIT     #D$IBRK,D.FLAG  ;Did an instruction breakpoint occur?
      69 001162  001406                           BEQ     2$              ;Br if not
      70 001164  162737  000002  000000G          SUB     #2,D.R7         ;Backup PC to point to BPT instruction
      71 001172  052737  000000G 000000G          BIS     #D$IPND,D.FLAG  ;Remember instruction must be executed
      72                                ;
      73                                ;  See if a proceed repeat count is in effect
      74                                ;
      75 001200  005737  000000G       2$:        TST     D.PCNT          ;Is a proceed repeat count in effect?
      76 001204  001407                           BEQ     3$              ;Br if not
      77 001206  032737  000000G 000000G          BIT     #D$FBRK,D.FLAG  ;Is this a forced breakpoint?
      78 001214  001003                           BNE     3$              ;Br if yes -- always stop
      79 001216  005337  000000G                  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  000000G 000000G 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
```

```
                                        .SBTTL  EXIT   -- Exit from debugger to user
                                ;-----------------------------------------------------------------------
                                ;  Exit from debugger to user's program.
                                ;
001246  042737  0000000 0000000 EXIT1:  BIC     #D$FBRK,D.FLAG  ;Turn off forced-breakpoint flag
001254  032737  000000C 0000000         BIT     #<D$SSTP!D$TSTP!D$DMON>,D.FLAG ;Are we single stepping?
001262  001403                          BEQ     3$              ;Br if not
001264  042737  0000000 0000000         BIC     #D$IPND,D.FLAG  ;Turn of instruction-pending if single step
001272  005037  0000000         3$:     CLR     D.LOC           ;Say no location is open
001276  042737  0000000 0000000         BIC     #D$RUN,D.FLAG   ;Say debugger is no longer running
001304  004737  007474'                  CALL    DBGOFF          ;Turn off terminal debug mode
001310  032737  000000C 0000000         BIT     #<D$IPND!D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Should we single step?
001316  001003                          BNE     4$              ;Br if yes
001320  004737  004340'                  CALL    BRKSET          ;Set breakpoints
001324  000406                          BR      1$
                                ;
                                ;  We are single stepping the program.
                                ;  Set the trace-trap flag in the user's TRAP instruction PSW so we
                                ;  will continue to have program control during processing of TRAP
                                ;  instructions.
                                ;
001326  106537  000036          4$:     MFPD    @#36            ;Get user's TRAP PSW
001332  052716  000020                  BIS     #TRCTRP,(SP)    ;Set the trace-trap flag in the PSW
001336  106637  000036                  MTPD    @#36            ;Store new TRAP PSW
                                ;
                                ;  If we are doing data monitoring, save current value of monitored cell
                                ;
001342  032737  0000000 0000000 1$:     BIT     #D$DMON,D.FLAG  ;Are we doing data cell monitoring?
001350  001411                          BEQ     2$              ;Br if not
001352  013705  0000000                 MOV     D.DADR,R5       ;Get address of cell being monitored
001356  106515                          MFPD    (R5)            ;Get current value from cell
001360  013704  0000000                 MOV     D.MASK,R4       ;Get current data mask
001364  005104                          COM     R4              ;Complement
001366  040405                          BIC     R4,R5           ;Clear all but bits of interest
001370  012637  0000000                 MOV     (SP)+,D.DOLD    ;Save old value
                                ;
                                ;  Restore user's registers
                                ;
001374  013746  0000000         2$:     MOV     D.R6,-(SP)
001400  106606                          MTPD    SP
001402  013705  0000000                 MOV     D.R5,R5
001406  013704  0000000                 MOV     D.R4,R4
001412  013703  0000000                 MOV     D.R3,R3
001416  013702  0000000                 MOV     D.R2,R2
001422  013701  0000000                 MOV     D.R1,R1
                                ;
                                ;  Exit point used if we did not enter debugger
                                ;
001426  013700  0000000         EXIT2:  MOV     D.R0,R0
                                ;
                                ;  See if we need to set Trap flag in PS
                                ;
001432  042737  000020  0000000         BIC     #TRCTRP,D.PS    ;Reset trace trap flag in PS
001440  032737  000000C 0000000         BIT     #<D$IPND!D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Do we want single step?
001446  001403                          BEQ     1$              ;Br if not
001450  052737  000020  0000000         BIS     #TRCTRP,D.PS    ;Set Trap flag in user's PS
001456  052737  000000C 0000000 1$:     BIS     #UMODE!UPMODE,D.PS ;Make sure user-mode is set in PS
```

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

```
                                        .SBTTL  NEWCMD -- Get new command
                                ;-----------------------------------------------------------------------
                                ;  Print carriage-return, line-feed and get new command
                                ;
001476                          NEWLIN: TPRINT  #CRLF
                                ;
                                ;  Get new command
                                ;
001506                          NEWCMD: PRINT   <DBG:>
                                ;
                                ;  Assume neither value1 nor value2 is specified
                                ;
001520  105037  0000000         GETCMD: CLRB    D.V1FL          ;Value1 not specified
001524  005037  0000000                 CLR     D.VAL1
001530  105037  0000000                 CLRB    D.V2FL          ;Value2 not specified
001534  005037  0000000                 CLR     D.VAL2
                                ;
                                ;  See if Value1 is specified
                                ;
001540  004737  005252'                 CALL    ACRVAL          ;Try to accrue value1
001544  010037  0000000                 MOV     R0,D.VAL1       ;Save value for value1
001550  110137  0000000                 MOVB    R1,D.V1FL       ;Remember if value1 specified
                                ;
                                ;  See if value2 is specified
                                ;
001554  004737  005720'                 CALL    GETCHR          ;Get delimiter
001560  120027  000073                  CMPB    R0,#';          ;is there a value2?
001564  001403                          BEQ     1$              ;Br if possibly yes
001566  004737  005764'                 CALL    PSHCHR          ;Save command character
001572  000406                          BR      2$
001574  004737  005252'         1$:     CALL    ACRVAL          ;Get value2
001600  010037  0000000                 MOV     R0,D.VAL2       ;Save value2
001604  110137  0000000                 MOVB    R1,D.V2FL       ;Save info about value2
                                ;
                                ;  Look up command character
                                ;
001610  004737  005720'         2$:     CALL    GETCHR          ;Get command character
001614  012701  000023                  MOV     #NUMCMD,R1      ;Get # valid command characters
001620  120061  000270'         3$:     CMPB    R0,CMDTBL(R1)   ;Search for character in table
001624  001406                          BEQ     4$              ;Br if found command character
001626  005301                          DEC     R1              ;More command chars to check?
001630  002373                          BGE     3$              ;Loop if yes
001632                                  ERR     #EM$IVC         ;Invalid command character
                                ;
                                ;  Found command character.
                                ;  Jump off to processing routine.
                                ;
001642  006301                  4$:     ASL     R1              ;Convert command index into word table index
001644  000171  000314'                 JMP     @CMDVEC(R1)     ;Jump to processing routine
```

```
                                        .SBTTL  -- Command processing routines --
                                        .SBTTL  "/"  -- Examine word
                                ;-------------------------------------------------------------------
                                ;  "address/" -- Examine contents of word
                                ;
001650  004737  006370'         CMDSLH: CALL    SETLOC          ;Set value1 as current location
001654  032737  000001  000000G CMDSL1: BIT     #1,D.LOC        ;Is location odd?
001662  001021                         BNE     CMDBKS          ;If yes then treat "/" like "\"
001664  105037  000000G                CLRB    D.BYTM          ;Say we are in word mode
001670  012737  000002  000000G        MOV     #2,D.ILEN       ;Say instruction length = 2 bytes
001676  004737  006614'               CALL    GETVAL          ;Get contents of location
001702  010037  000000G                MOV     R0,D.LVAL       ;Save last value displayed
001706  004737  006100'               CALL    PRTWRD          ;Print the value
001712                                 PRINT   <  >            ;Print two spaces
001722  000137  001520'               JMP     GETCMD          ;Go get next command

                                        .SBTTL  "\"  -- Examine byte
                                ;-------------------------------------------------------------------
                                ;  "address\" -- Examine contents of a byte
                                ;
001726  004737  006370'         CMDBKS: CALL    SETLOC          ;Set value1 as current location
001732  112737  000001  000000G CMDBK1: MOVB    #1,D.BYTM       ;Say we are operating in byte mode
001740  112737  000001  000000G        MOVB    #1,D.ILEN       ;Say instruction length = 1 byte
001746  004737  006614'               CALL    GETVAL          ;Get contents of location
001752  010037  000000G                MOV     R0,D.LVAL       ;Save last value displayed
001756  010002                         MOV     R0,R2           ;Save value
001760  004737  006154'               CALL    PRTBYT          ;Print value
001764                                 PRINT   < = >           ;Put in equal sign
001774  120227  000040                 CMPB    R2,#40          ;Is this a printing character?
002000  103002                         BHIS    1$              ;Br if yes
002002  112702  000056                 MOVB    #'.,R2          ;Use period for non-printing chars
002006                          1$:    .TTYOUT R2              ;Print character
002014  005237  000000G                INC     D.PCOL          ;Advance print column
002020                                 PRINT   <  >            ;Print 2 spaces
002030  000137  001520'               JMP     GETCMD          ;Go get next command
```

```
                        .SBTTL  "[" -- Decode value as an instruction
                ;-----------------------------------------------------------------------
                ;  Open a cell and display its contents as an instruction.
                ;
002034          CMDDCD:
                ;
                ;  See if an address was specified
                ;
002034  105737  0000000         TSTB    D.V1FL          ;Was an address specified?
002040  001402                  BEQ     1$              ;Br if not
002042  004737  006370'         CALL    SETLOC          ;Set address of currently open cell
002046  013701  0000000   1$:   MOV     D.LOC,R1        ;Is a cell currently open?
002052  001002                  BNE     2$              ;Br if yes
002054  000137  001476'         JMP     NEWLIN          ;Nothing to decode
002060  032701  000001    2$:   BIT     #1,R1           ;Is the address odd?
002064  001402                  BEQ     3$              ;Br if not
002066  000137  001726'         JMP     CMDBKS          ;Treat like "\" if odd
002072  105737  0000000   3$:   TSTB    D.LOCM          ;Is cell internal or external?
002076  002002                  BGE     4$              ;Br if external
002100  000137  001654'         JMP     CMDSL1          ;Treat [ like / for internal cell
                ;
                ;  Decode the instruction
                ;
002104  004737  007510'   4$:   CALL    DECODE          ;Decode the instruction
                ;
                ;  Tab over some
                ;
002110  013700  0000000         MOV     D.PCOL,R0       ;Get current print column
002114  062700  000010          ADD     #8.,R0          ;Tab over some
002120  042700  000007          BIC     #7,R0
002124  004737  011734'         CALL    ODCTAB          ;Tab over
                ;
                ;  Finished
                ;
002130  000137  001520'         JMP     GETCMD
```

```
                                        .SBTTL  "@"  -- Open indirect
                                ;-------------------------------------------------------------------
                                ;  Open the cell whose address is contained in the cell pointed to
                                ;  by D.LOC.
                                ;
002134  013702  0000000         CMDAT:  MOV     D.LVAL,R2       ;Get last displayed value
002140  010237  0000000         CMDAT1: MOV     R2,D.LOC        ;Set as new location to display
002144  112737  000001  0000000        MOVB    #1,D.LOCM       ;Say new address is in user's program
002152                                  TPRINT  #CRLF           ;Go to a new line
                                ;
                                ;  See if we should do the display in byte or word mode
                                ;
002162  032702  000001                  BIT     #1,R2           ;Should we display in byte or word mode?
002166  001010                         BNE     1$              ;Br if need to go to byte mode
                                ;
                                ;  Display new location in word mode
                                ;
002170  004737  006750'                 CALL    SHOLOC          ;Display the current address
002174                                  PRINT   </>             ;Display "/"
002204  000137  001654'                 JMP     CMDSL1          ;Simulate the "/" command
                                ;
                                ;  Display in byte mode
                                ;
002210  004737  006750'         1$:     CALL    SHOLOC          ;Display the current address
002214                                  PRINT   <\>             ;Display "\"
002224  000137  001732'                 JMP     CMDBK1          ;Simulate the "\" command

                                        .SBTTL  "_"  -- Open cell indirect, relative to PC
                                ;-------------------------------------------------------------------
                                ;  "_" -- Open indirect relative to PC.
                                ;
002230  013702  0000000         CMDUS:  MOV     D.LVAL,R2       ;Get last displayed value
002234  063702  0000000                 ADD     D.LOC,R2        ;Add PC of instruction
002240  062702  000002                  ADD     #2,R2           ;Plus 2
002244  000137  002140'                 JMP     CMDAT1          ;Now treat like "@" command
```

```
                                        .SBTTL  "!"  -- Convert address to relocation offset
                                ;-------------------------------------------------------------------
                                ;  "n!" -- Convert current address to relocation offset
                                ;
002250                          CMDEXP:
                                ;
                                ;  If no relocation register number was provided, find closest relocation
                                ;  base.
                                ;
002250  013701  0000000                 MOV     D.VAL1,R1       ;Get relocation register #
002254  105737  0000000                 TSTB    D.V1FL          ;Was a relocation register specified?
002260  003422                          BLE     2$              ;Br if not
                                ;
                                ;  A relocation register number was provided.
                                ;  Display the relocation register number and the offset.
                                ;
002262                                  PRINT   <=>             ;Print "="
002272  010100                          MOV     R1,R0           ;Get the relocation register number
002274  004737  005772'                 CALL    PRTOCT          ;Print the relocation reg #
002300                                  PRINT   <,>             ;Print ","
002306  006301                          ASL     R1              ;Convert relocation # to word table index
002310  013700  0000000                 MOV     D.LOC,R0        ;Get original value
002314  166100  0000000                 SUB     D.RLBS(R1),R0   ;Subtract relocation base
002320  004737  005772'                 CALL    PRTOCT          ;Print the offset
002324  000404                          BR      8$
                                ;
                                ;  No relocation register number was specified
                                ;
002326  013700  0000000         2$:     MOV     D.LOC,R0        ;Get address
002332  004737  007172'                 CALL    SHOADR          ;Display it with best relocation base
002336                          8$:     PRINT   <  >            ;Print 2 spaces
                                ;
                                ;  Finished
                                ;
002346  000137  001520'         9$:     JMP     GETCMD          ;Go get the next command
```

```
                                        .SBTTL  "R"  -- Set relocation base register
                                ;-----------------------------------------------------------------------
                                ;  "address;nR" -- Set relocation base register
                                ;
002352                          CMDR:
                                ;
                                ;  If no value was specified for argument 2, then we are being
                                ;  asked to calculate the difference between the currently displayed
                                ;  value and the base of the relocation area.
                                ;  If argument 2 was specified, then we are setting the base of a
                                ;  relocation area.
                                ;
002352  105737  0000000                 TSTB    D.V2FL          ;Was argument 2 specified?
002356  001417                          BEQ     2$              ;Br if not
                                ;
                                ;  Set base address for relocation area
                                ;
002360  013701  0000000                 MOV     D.VAL2,R1       ;Get register number
002364  020127  000007                  CMP     R1,#7           ;Should not exceed 7
002370  101404                          BLOS    1$              ;Br if ok
002372                                  ERR     #EM$IVL
002402  006301                  1$:     ASL     R1              ;Convert to word table index
002404  013761  0000000 0000000         MOV     D.VAL1,D.RLBS(R1);Set address for relocation base
002412  000137  001476'                 JMP     NEWLIN          ;Go get next command
                                ;
                                ;  Calculate offset relative to relocation base
                                ;
002416                          2$:     PRINT   <=>             ;Print "="
002426  105737  0000000                 TSTB    D.V1FL          ;Was a relocation reg # provided?
002432  001436                          BEQ     4$              ;Br if not
002434  013701  0000000                 MOV     D.VAL1,R1       ;Get register number
002440  020127  000007                  CMP     R1,#7           ;Should not exceed 7
002444  101404                          BLOS    3$              ;Br if ok
002446                                  ERR     #EM$IVL
002456  006301                  3$:     ASL     R1              ;Convert to word table index
002460  013700  0000000                 MOV     D.LVAL,R0       ;Get currently open value
002464  166100  0000000                 SUB     D.RLBS(R1),R0   ;Calculate offset
002470  010002                          MOV     R0,R2           ;Hold the offset value
002472  010100                          MOV     R1,R0           ;Get offset region #
002474  006200                          ASR     R0              ;Convert to ordinal number
002476  004737  005772'                 CALL    PRTOCT          ;Print relocation #
002502                                  PRINT   <,>             ;Print comma
002510  010200                          MOV     R2,R0           ;Get offset value
002512  004737  005772'                 CALL    PRTOCT          ;Print it
002516                                  PRINT   <  >            ;Print a space
002526  000410                          BR      9$
                                ;
                                ;  No relocation register number was provided
                                ;  Show address with best relocation
                                ;
002530  013700  0000000         4$:     MOV     D.LVAL,R0       ;Get value of interest
002534  004737  007172'                 CALL    SHOADR          ;Display the address
002540                                  PRINT   <  >            ;Print 2 spaces
                                ;
                                ;  Finished
                                ;
002550  000137  001520'         9$:     JMP     GETCMD          ;Go get next command
```

```
      1                                         .SBTTL  "B"  -- Set breakpoint
      2                                 ;-----------------------------------------------------------------------
      3                                 ;  "address;nB" -- Set breakpoint
      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
```

```
                                        .SBTTL  "G"  -- Start program execution
                                ;-------------------------------------------------------------
                                ;  "address;G" -- Start program execution.
                                ;
002656  105737  0000000         CMDG:   TSTB    D.V1FL          ;Was a starting address specified?
002662  003403                  1$:     BLE     1$              ;Br if not
002664  013737  0000000 0000000         MOV     D.VAL1,D.R7     ;Set starting address
002672                          1$:     TPRINT  #CRLF           ;Go to new line
002702  005037  0000000                 CLR     D.PCNT          ;Clear any proceed count
002706  000137  001246'                 JMP     EXIT1           ;Start running program

                                        .SBTTL  "P"  -- Proceed from breakpoint
                                ;-------------------------------------------------------------
                                ;  "n;P" -- Proceed from breakpoint
                                ;
002712  105737  0000000         CMDP:   TSTB    D.V1FL          ;Was a proceed repeat count specified?
002716  003403                          BLE     1$              ;Br if not
002720  013737  0000000 0000000         MOV     D.VAL1,D.PCNT   ;Set proceed repeat count
002726                          1$:     TPRINT  #CRLF           ;Go to new line
002736  000137  001246'                 JMP     EXIT1           ;Enter user's program

                                        .SBTTL  "S"  -- Set/Reset single-step mode
                                ;-------------------------------------------------------------
                                ;  ";nS" -- Set or reset single-step mode
                                ;
002742  042737  0000000 0000000 CMDS:   BIC     #D$SSTP,D.FLAG  ;Assume he wants to reset single-step mode
002750  105737  0000000                 TSTB    D.V2FL          ;Was a value specified for "n"?
002754  001406                          BEQ     1$              ;Br if not
002756  005737  0000000                 TST     D.VAL2          ;Is value 2 non-zero?
002762  001403                          BEQ     1$              ;If zero, don't single step
002764  052737  0000000 0000000         BIS     #D$SSTP,D.FLAG  ;Set single-step mode
002772  000137  001476'         1$:     JMP     NEWLIN          ;Go get next command
```

```
                                        .SBTTL  "TAB"-- Single step the program
                                ;-----------------------------------------------------------------
                                ;  The TAB command causes us to go into temporary single step mode
                                ;  and to execute the next instruction.
                                ;  If "valueTAB is specified, we use the value as the address of
                                ;  the next instruction to execute.
                                ;  If "OTAB" is specified, we step over a subprogram call.
                                ;
002776                          CMDTAB:
                                ;
                                ;  See if a value was specified with TAB
                                ;
002776  105737  0000000                 TSTB    D.V1FL          ;Was value 1 specified?
003002  001407                          BEQ     2$              ;Br if not
003004  105037  0000000                 CLRB    D.V1FL          ;Clear value-1 flag so it won't confuse ;P
                                ;
                                ;  A value was specified with TAB.
                                ;  If 0 was specified, skip over a subprogram call.
                                ;  If a non-zero value was specified, use this as a starting address.
                                ;
003010  013700  0000000         1$:     MOV     D.VAL1,R0       ;Get specified value
003014  001407                          BEQ     5$              ;0 ==> Skip over subprogram call
                                ;
                                ;  A starting address was specified.
                                ;
003016  010037  0000000                 MOV     R0,D.R7         ;Set starting address
                                ;
                                ;  Single step the program.
                                ;
003022  052737  0000000 0000000 2$:     BIS     #D$TSTP,D.FLAG  ;Say we are in temporary single step mode
003030  000137  002712'                 JMP     CMDP            ;Now treat tab like ;P
                                ;
                                ;  Argument value 1 is 0 (zero).
                                ;  Step over a subprogram call.
                                ;  Make sure the current instruction is a CALL.
                                ;
003034  013701  0000000         5$:     MOV     D.R7,R1         ;Get address of current instruction
003040  106521                          MFPD    (R1)+           ;Get instruction from user's program
003042  012602                          MOV     (SP)+,R2        ;Get the instruction from the stack
003044  010200                          MOV     R2,R0           ;Copy the instruction word
003046  042700  000777                  BIC     #777,R0         ;Clear register # and address field
003052  020027  004000                  CMP     R0,#0004000     ;Is this a JSR instruction?
003056  001361                          BNE     2$              ;If not, ignore the number in front of TAB
                                ;
                                ;  Determine address of instruction beyond CALL and set it as a location
                                ;  where a temporary breakpoint is to be set.
                                ;
003060  010200                          MOV     R2,R0           ;Get the instruction
003062  042700  177770                  BIC     #^C7,R0         ;Clear all but addressing register #
003066  020027  000007                  CMP     R0,#7           ;Is it using register 7 (PC)?
003072  001003                          BNE     3$              ;Br if not
003074  062701  000002                  ADD     #2,R1           ;2 bytes for subroutine address if PC used
003100  000410                          BR      4$
003102  010200                  3$:     MOV     R2,R0           ;Get back the instruction
003104  042700  177707                  BIC     #^C70,R0        ;Clear all but addressing mode
003110  072027  177775                  ASH     #-3,R0          ;Right justify the addressing mode
003114  116000  003156'                 MOVB    INSLEN(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
```

```
                                        .SBTTL  "X"  -- Display RAD50 value
                                ;-------------------------------------------------------------
                                ;  "X" -- Convert currently displayed value from RAD50 to ascii.
                                ;
003166                          CMDX:   PRINT   <=>             ;Print "="
003176  013700  0000000                 MOV     D.LVAL,R0       ;Get last displayed value
003202  004737  006232'                 CALL    PRTR50          ;Convert to ascii and print
003206                                  PRINT   <  >            ;Print spaces
003216  012737  000002  0000000         MOV     #2,D.ILEN       ;Say instruction length = 2 bytes
                                ;
                                ;  See if user wants to enter a new RAD50 value for this cell
                                ;
003224  004737  005720'                 CALL    GETCHR          ;Get 1st character
003230  004737  005764'                 CALL    PSHCHR          ;Push the character
003234  120027  000015                  CMPB    R0,#CR          ;No new value wanted?
003240  001454                          BEQ     7$              ;Br if no new value specified
003242  120027  000012                  CMPB    R0,#LF          ;Advance to next cell?
003246  001451                          BEQ     7$              ;Br if yes
                                ;
                                ;  Now accept a new RAD50 value
                                ;
003250  005001                          CLR     R1              ;Form new RAD50 value in R1
003252  012702  000003                  MOV     #3,R2           ;Set count of # chars to accrue
003256  004737  005720'         1$:     CALL    GETCHR          ;Get next character of value
003262  120027  000015                  CMPB    R0,#CR          ;Reached end?
003266  001434                          BEQ     2$              ;Br if yes
003270  120027  000012                  CMPB    R0,#LF          ;LF is also end
003274  001431                          BEQ     2$
003276  012703  000220'                 MOV     #R50CHR,R3      ;Point to RAD50 character table
003302  120023                  3$:     CMPB    R0,(R3)+        ;Search for char in table
003304  001407                          BEQ     4$              ;Br if found
003306  020327  000270'                 CMP     R3,#R50CHR+50   ;Checked all of table?
003312  103773                          BLO     3$              ;Loop if not
003314                                  ERR     #EM$IRC         ;Invalid RAD50 character
003324  162703  000221'         4$:     SUB     #R50CHR+1,R3    ;Get RAD50 character value
003330  070127  000050                  MUL     #50,R1          ;Multiply previous value by 50
003334  060301                          ADD     R3,R1           ;Add new character value
003336  077231                          SOB     R2,1$           ;Loop if more characters needed
003340  004737  005720'         6$:     CALL    GETCHR          ;Ignore other characters up to CR
003344  120027  000015                  CMPB    R0,#CR
003350  001403                          BEQ     2$
003352  120027  000012                  CMPB    R0,#LF
003356  001370                          BNE     6$
                                ;
                                ;  We have gotten the RAD50 value.  Store it in currently open cell
                                ;
003360  004737  005764'         2$:     CALL    PSHCHR          ;Save the terminating character
003364  010100                          MOV     R1,R0           ;Get value to R0 for STRVAL
003366  004737  006412'                 CALL    STRVAL          ;Store value into currently open cell
                                ;
                                ;  If command ended with Line feed, open next cell
                                ;
003372  005037  0000000         7$:     CLR     D.PCOL          ;Say print column = 0
003376  004737  005720'                 CALL    GETCHR          ;Get the terminating character
003402  120027  000012                  CMPB    R0,#LF          ;Open the next cell?
003406  001004                          BNE     9$              ;Br if not
003410  105037  0000000                 CLRB    D.V1FL          ;Tell LF not to store into current cell
```

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

```
                                        .SBTTL  "M"  -- Monitor data cell
                                ;-----------------------------------------------------------------------
                                ;  "address;valueM" -- Monitor data cell
                                ;
003424  042737  000000C 000000G CMDM:   BIC     #<D$DMON!D$DVAL>,D.FLAG ;Reset data-cell monitoring flags
003432  005737  0000000          TST     D.VAL1          ;Was an address specified?
003436  001420                   BEQ     9$              ;Br if not -- Disable monitoring
003440  052737  0000000 000000G        BIS     #D$DMON,D.FLAG  ;Enable data monitoring
003446  013700  0000000          MOV     D.VAL1,R0       ;Get address of cell to monitor
003452  010037  0000000          MOV     R0,D.DADR       ;Set it as control address
                                ;
                                ;  See if breakpoint is to be triggered whenever value in cell changes
                                ;  or only when a specified value occurs.
                                ;
003456  105737  0000000          TSTB    D.V2FL          ;Was a value specified?
003462  003406                   BLE     9$              ;Br if not
003464  052737  0000000 000000G        BIS     #D$DVAL,D.FLAG  ;Watch for specified value
003472  013737  0000000 000000G        MOV     D.VAL2,D.DTRG   ;Save the specified value
                                ;
                                ;  Finished
                                ;
003500  000137  001476'          9$:     JMP     NEWLIN          ;Go get next command
```

```
                                        .SBTTL  "CR" -- Close current location
                                ;-------------------------------------------------------------
                                ;  "carriage-return" -- Close current cell
                                ;
003504  005037  000000  CMDCR:  CLR     D.PCOL          ;Say print column = 0
003510  105737  000000          TSTB    D.V1FL          ;Was a new value specified for this cell?
003514  003404                  BLE     9$              ;Br if not
003516  013700  000000          MOV     D.VAL1,R0       ;Yes, get the value
003522  004737  006412'          CALL    STRVAL          ;Store value into current location
003526  005037  000000  9$:     CLR     D.LOC           ;Say no location open now
003532  000137  001506'          JMP     NEWCMD          ;Go get next command

                                        .SBTTL  "LF" -- Close location and advance to next
                                ;-------------------------------------------------------------
                                ;  "line-feed" -- Close current cell and advance to next
                                ;
003536  005037  000000  CMDLF:  CLR     D.PCOL          ;Say print column = 0
003542  005737  000000          TST     D.LOC           ;Is a location open now?
003546  001447                  BEQ     5$              ;Br if not
003550  105737  000000          TSTB    D.V1FL          ;Was a new value specified for this cell?
003554  003404                  BLE     1$              ;Br if not
003556  013700  000000          MOV     D.VAL1,R0       ;Get new value
003562  004737  006412'          CALL    STRVAL          ;Store into currently open cell
003566  105737  000000  1$:     TSTB    D.BYTM          ;Are we in byte or word mode?
003572  001017                  BNE     2$              ;Br if in byte mode
003574  062737  000002  000000  ADD     #2,D.LOC        ;Advance location counter
003602                          .TTYOUT #CR             ;Go to new line
003612  004737  006750'          CALL    SHOLOC          ;Display address of new location
003616                          PRINT   < / >           ;Display "/"
003626  000137  001654'          JMP     CMDSL1          ;Simulate "/" command
003632  005237  000000  2$:     INC     D.LOC           ;Advance to next byte
003636                          TPRINT  #CRCHAR         ;go to new line
003646  004737  006750'          CALL    SHOLOC          ;Display address of new location
003652                          PRINT   < \ >           ;Display "\"
003662  000137  001732'          JMP     CMDBK1          ;Simulate "\" command
003666  000137  001476'  5$:     JMP     NEWLIN          ;No cell open -- Ignore LF
```

```
                                        .SBTTL  "]"  -- Close location and advance to instruction
                                ;-----------------------------------------------------------------
                                ;  "]" -- Close the current location and open next as an instruction.
                                ;
003672  005037  0000000         CMDRSB: CLR     D.PCOL          ;Say print column = 0
003676  105737  0000000                 TSTB    D.V1FL          ;Was an address specified?
003702  001402                         BEQ     3$              ;Br if not
003704  000137  002034'                 JMP     CMDDCD          ;Treat ] like [
003710  005737  0000000         3$:     TST     D.LOC           ;Is a location open now?
003714  001440                         BEQ     5$              ;Br if not
003716  105737  0000000                 TSTB    D.BYTM          ;Are we in byte or word mode?
003722  001017                         BNE     2$              ;Br if in byte mode
003724  063737  0000000 0000000                 ADD     D.ILEN,D.LOC    ;Advance location counter
003732                                  TPRINT  #CRLF           ;Go to new line
003742  004737  006750'                 CALL    SHOLOC          ;Display address of new location
003746                                  PRINT   <[>             ;Display "["
003756  000137  002034'                 JMP     CMDDCD          ;Simulate "[" command
003762  005237  0000000         2$:     INC     D.LOC           ;Advance to next byte
003766                                  TPRINT  #CRCHAR         ;go to new line
003776  004737  006750'                 CALL    SHOLOC          ;Display address of new location
004002                                  PRINT   <\>             ;Display "\"
004012  000137  001726'                 JMP     CMDBKS          ;Simulate "\" command
004016  000137  001476'         5$:     JMP     NEWLIN          ;No cell open -- Ignore LF
```

```
                                        .SBTTL  "^"  -- Close current location and go to previous
                                ;-----------------------------------------------------------------------
                                ;  "Backspace" -- Equivalent to up-arrow
                                ;
004022                          CMDBSP: .TTYOUT #136            ;Print "^"
004032  000400                          BR      CMDUP           ;Now treat backspace like up-arrow
                                ;-----------------------------------------------------------------------
                                ;  "^" -- Close current location and go to previous one.
                                ;
004034  005037  000000G         CMDUP:  CLR     D.PCOL          ;Say print column = 0
004040  005737  000000G                 TST     D.LOC           ;Is a cell open?
004044  001002                          BNE     3$              ;Br if yes
004046  000137  001476'                 JMP     NEWLIN          ;Ignore if no cell open
004052  105737  000000G         3$:     TSTB    D.V1FL          ;Was a new value specified for this cell?
004056  003404                          BLE     1$              ;Br if not
004060  013700  000000G                 MOV     D.VAL1,R0       ;Get new value
004064  004737  006412'                 CALL    STRVAL          ;Store into currently open cell
004070  105737  000000G         1$:     TSTB    D.BYTM          ;Are we in byte or word mode?
004074  001017                          BNE     2$              ;Br if in byte mode
004076  162737  000002  000000G        SUB     #2,D.LOC        ;Move to previous word
004104                                  TPRINT  #CRLF           ;Go to new line
004114  004737  006750'                 CALL    SHOLOC          ;Display address of new location
004120                                  PRINT   </>             ;Display "/"
004130  000137  001654'                 JMP     CMDSL1          ;Simulate "/" command
004134  005337  000000G         2$:     DEC     D.LOC           ;Move to previous byte
004140                                  TPRINT  #CRLF           ;go to new line
004150  004737  006750'                 CALL    SHOLOC          ;Display address of new location
004154                                  PRINT   <\>             ;Display "\"
004164  000137  001732'                 JMP     CMDBK1          ;Simulate "\" command
```

```
                                        .SBTTL  -- Subroutines --
                                        .SBTTL  DBGINI -- Initialize the debugger
                                ;-----------------------------------------------------------------------
                                ;  DBGINI is called to initialize the debugger.
                                ;  The debugger greeting message is also printed.
                                ;
004170  010046                 DBGINI: MOV     R0,-(SP)
                                ;
                                ;  Initialize some debugger values
                                ;
004172  012700  0000000                 MOV     #D.START,R0     ;Point to start of debugging data
004176  005020                  1$:     CLR     (R0)+           ;Clear all debugging data to 0
004200  020027  0000000                 CMP     R0,#D.END       ;Finished?
004204  103774                         BLO     1$              ;Loop if not
004206  012737  177777  0000000         MOV     #177777,D.MASK  ;Initialize the data mask
004214  052737  0000000 0000000         BIS     #D$INIT,D.FLAG  ;Say initialization has been done
                                ;
                                ;  Set previous-mode = user in our PSW so we can use MFPD/MTPD to
                                ;  access user's program.
                                ;
004222  052737  0000000 0000000         BIS     #UPMODE,@#PSW   ;Say previous mode = user
                                ;
                                ;  Print greeting message
                                ;
004230                                  TPRINT  #TM$GRT         ;Print greeting message
                                ;
                                ;  Finished
                                ;
004240  012600                  MOV     (SP)+,R0
004242  000207                  RETURN
```

```
                                        .SBTTL  BRKRST -- Reset breakpoint instructions
                                ;-----------------------------------------------------------------------
                                ;  BRKRST is called on entry to the debugger when a breakpoint occurs.
                                ;  It restores the contents of the instruction cells that were replaced
                                ;  by BPT instructions for breakpoints.
                                ;
004244  010146                  BRKRST: MOV     R1,-(SP)
004246  032737  0000000 0000000         BIT     #D$BKST,D.FLAG  ;Are breakpoints set in program?
004254  001427                          BEQ     9$              ;Br if not
004256  032737  0000000 0000000         BIT     #D$CKBK,D.FLAG  ;Are there any active instruction breaks?
004264  001423                          BEQ     9$              ;Br if not
                                ;
                                ;  Reset instructions that were overwritten by breakpoints
                                ;
004266  005001                          CLR     R1              ;Init index into breakpoint tables
004270  016100  0000000         1$:     MOV     D.BKAD(R1),R0   ;Get address of breakpoint
004274  001407                          BEQ     2$              ;Br if this breakpoint not in use
004276  106510                          MFPD    (R0)            ;Get breakpoint instruction
004300  022627  000003                  CMP     (SP)+,#3        ;Is this a breakpoint instruction?
004304  001003                          BNE     2$              ;Br if not
004306  016146  0000000                 MOV     D.BKSV(R1),-(SP);Get saved instruction
004312  106610                          MTPD    (R0)            ;Restore saved instruction
004314  062701  000002          2$:     ADD     #2,R1           ;Advance breakpoint index
004320  020127  000020                  CMP     R1,#16.         ;Checked all breakpoints?
004324  101761                          BLOS    1$              ;Loop if more to check
004326  042737  0000000 0000000         BIC     #D$BKST,D.FLAG  ;Say breakpoints no longer set
                                ;
                                ;  Finished
                                ;
004334  012601                  9$:     MOV     (SP)+,R1
004336  000207                          RETURN
```

```
                                        .SBTTL  BRKSET -- Set breakpoints
                                ;-------------------------------------------------------------------
                                ;  BRKSET is called when leaving the debugger and entering the user's program.
                                ;  It sets BPT instructions in those locations marked for breakpoints.
                                ;
004340  010146                  BRKSET: MOV     R1,-(SP)
004342  032737  0000000 0000000         BIT     #D$CKBK,D.FLAG  ;Are there any instruction breakpoints?
004350  001416                          BEQ     3$              ;Br if not
                                ;
                                ;  Set breakpoints
                                ;
004352  012701  000020                  MOV     #16.,R1         ;Get index for last breakpoint
004356  016100  0000000          1$:     MOV     D.BKAD(R1),R0   ;Get address where breakpoint goes
004362  001406                          BEQ     2$              ;Br if this breakpoint not in use
004364  106510                          MFPD    (R0)            ;Get current contents of location
004366  012661  0000000                 MOV     (SP)+,D.BKSV(R1);Save original contents of cell
004372  012746  000003                  MOV     #3,-(SP)        ;Push BPT instruction
004376  106610                          MTPD    (R0)            ;Store BPT instruction into user's program
004400  162701  000002          2$:     SUB     #2,R1           ;More breakpoints to check?
004404  002364                          BGE     1$              ;Br if yes
004406  052737  0000000 0000000 3$:     BIS     #D$BKST,D.FLAG  ;Set flag saying breakpoints set in program
                                ;
                                ;  Finished
                                ;
004414  012601                          MOV     (SP)+,R1
004416  000207                          RETURN
```

```
                                        .SBTTL  BRKCHK -- Determine type of breakpoint
                                ;-----------------------------------------------------------------
                                ;  Determine the type of breakpoint that has occurred.
                                ;  There are four different types of breakpoints:
                                ;   1. Data breakpoints that are triggered when a monitored cell changes
                                ;      value or takes on a specified value.
                                ;   2. Instruction breakpoints set by use of the "a;nB" instruction.
                                ;   3. Instruction breakpoint used to catch a CALL return.
                                ;   4. Single-step breakpoints that result from the use of the ";1S"
                                ;      command.
                                ;
                                ;  Inputs:
                                ;   D.R7 = PC after the break
                                ;
                                ;  Outputs:
                                ;   D$DBRK in D.FLAG ==> Data breakpoint
                                ;   D$IBRK in D.FLAG ==> Instruction breakpoint
                                ;   D$SBRK in D.FLAG ==> Single-step breakpoint or CALL breakpoint
                                ;   D$FBRK in D.FLAG ==> Forced breakpoint (user typed ctrl-B)
                                ;   D.BKNM = Breakpoint number
                                ;
004420  010146                  BRKCHK: MOV     R1,-(SP)
                                ;
                                ;  Clear instruction and data breakpoint flags
                                ;
004422  042737  000000C 000000G         BIC     #<D$DBRK!D$IBRK!D$SBRK>,D.FLAG ;No breakpoints yet
                                ;
                                ;  See if a data breakpoint is active
                                ;
004430  032737  0000000 000000G         BIT     #D$DMON,D.FLAG  ;Is a data breakpoint active?
004436  001425                          BEQ     1$              ;Br if not
004440  013701  0000000                 MOV     D.DADR,R1       ;Get address of monitored cell
004444  106511                          MFPD    (R1)            ;Get contents of monitored cell
004446  013701  0000000                 MOV     D.MASK,R1       ;Get data mask
004452  005101                          COM     R1              ;Complement
004454  040116                          BIC     R1,(SP)         ;Leave only bits of interest
004456  032737  0000000 000000G         BIT     #D$DVAL,D.FLAG  ;Are we waiting for a specified value?
004464  001004                          BNE     2$              ;Br if yes
                                ;  Trigger data breakpoint whenever monitored data cell changes value
004466  022637  0000000                 CMP     (SP)+,D.DOLD    ;Has value changed?
004472  001004                          BNE     3$              ;Br if yes -- trigger breakpoint
004474  000406                          BR      1$              ;Hasn't changed -- no data break
                                ;  Trigger data breakpoint if value takes on specified value
004476  022637  0000000         2$:     CMP     (SP)+,D.DTRG    ;Does cell have value of interest?
004502  001003                          BNE     1$              ;Br if not -- no data breakpoint
                                ;
                                ;  Trigger a data breakpoint
                                ;
004504  052737  0000000 000000G 3$:     BIS     #D$DBRK,D.FLAG  ;Set data-breakpoint flag
                                ;
                                ;  See if we need to check for instruction or single-step breakpoints
                                ;
004512  032737  000000C 000000G 1$:     BIT     #<D$SSTP!D$TSTP!D$CKBK>,D.FLAG ;Any instruction breaks or stp?
004520  001442                          BEQ     9$              ;Br if neither instruction brks or single step
                                ;
                                ;  Ignore single-step and instruction breakpoints if we just
                                ;  executed a pending instruction.
```

```
                                        ;
004522  032737  0000000  0000000       BIT     #D$IPND,D.FLAG  ;Did we just execute a pending instruction?
004530  001036                         BNE     9$              ;Br if yes
                                        ;
                                        ;  See if this is a single-step breakpoint
                                        ;
004532  032737  0000000  0000000       BIT     #D$SSTP!D$TSTP,D.FLAG ;Are we single stepping program?
004540  001407                         BEQ     4$              ;Br if not
004542  052737  0000000  0000000 5$:    BIS     #D$SBRK,D.FLAG  ;Set single-step breakpoint flag
004550  112737  000011   0000000       MOVB    #9.,D.BKNM      ;Say this is breakpoint # 9
004556  000423                         BR      9$
                                        ;
                                        ;  See if this is an instruction breakpoint
                                        ;
004560  012701  000020           4$:    MOV     #16.,R1         ;Init breakpoint index
004564  013700  0000000                  MOV     D.R7,R0         ;Get address after breakpoint
004570  162700  000002                   SUB     #2,R0           ;Point to break instruction
004574  020061  0000000          7$:    CMP     R0,D.BKAD(R1)   ;Search for which breakpoint was hit
004600  001404                         BEQ     6$              ;Br if found it
004602  162701  000002                   SUB     #2,R1           ;Get next breakpoint index
004606  002372                         BGE     7$              ;Loop if more breakpoint to check
004610  000406                         BR      9$              ;This is not an instruction breakpoint
                                        ;
                                        ;  We hit an instruction breakpoint
                                        ;
004612  006201                   6$:    ASR     R1              ;Get breakpoint number
004614  110137  0000000                  MOVB    R1,D.BKNM
004620  052737  0000000  0000000       BIS     #D$IBRK,D.FLAG  ;Remember we had an instruction break
                                        ;
                                        ;  Finished
                                        ;
004626  012601                   9$:    MOV     (SP)+,R1
004630  000207                         RETURN
```

```
                                        .SBTTL  SHOBRK -- Display breakpoint information
                                ;-----------------------------------------------------------------
                                ;  SHOBRK is called to display information about a breakpoint
                                ;  that has been triggered.
                                ;
004632  010146                  SHOBRK: MOV     R1,-(SP)
004634                                  TPRINT  #CRLF           ;Get to start of new line
                                ;
                                ;  See if a data breakpoint occurred
                                ;
004644  032737  000000G 000000G         BIT     #D$DBRK,D.FLAG  ;Did a data breakpoint occur?
004652  001454                          BEQ     1$              ;Br if not
004654                                  PRINT   <Data break at >
004700  013700  000000G                 MOV     D.R7,R0         ;Get breakpoint PC
004704  004737  007146'                 CALL    LSTADR          ;Display the address
004710                                  PRINT   < -- >
004722  013700  000000G                 MOV     D.DADR,R0       ;Get address of monitored cell
004726  004737  007146'                 CALL    LSTADR          ;Display it
004732                                  PRINT   < = >
004742  013700  000000G                 MOV     D.DADR,R0       ;Get address of cell
004746  106510                          MFPD    (R0)            ;Get current contents of cell
004750  012600                          MOV     (SP)+,R0
004752  010037  000000G                 MOV     R0,D.DOLD       ;Save old cell value
004756  004737  006100'                 CALL    PRTWRD          ;Display value of cell
004762  013700  000000G                 MOV     D.MASK,R0       ;Get data mask
004766  005100                          COM     R0              ;Complement
004770  040037  000000G                 BIC     R0,D.DOLD       ;Clear all but bits of interest
004774                                  TPRINT  #CRLF           ;Go to new line
                                ;
                                ;  See if a single-step breakpoint occurred
                                ;
005004  032737  000000G 000000G 1$:     BIT     #D$SBRK,D.FLAG  ;Did a single-step breakpoint occur?
005012  001406                          BEQ     2$              ;Br if not
005014                          5$:     PRINT   <Step:>         ;Print heading
005026  000451                          BR      3$              ;Now treat like instruction breakpoing
                                ;
                                ;  See if an instruction breakpoint occurred
                                ;
005030  032737  000000G 000000G 2$:     BIT     #D$IBRK,D.FLAG  ;Instruction breakpoint?
005036  001426                          BEQ     4$              ;Br if not
005040                                  TPRINT  #CRLF           ;Go to new line
005050  123727  000000G 000010          CMPB    D.BKNM,#8.      ;CALL breakpoint?
005056  001003                          BNE     6$              ;Br if not
005060  005037  000000G                 CLR     D.CBRK          ;Clear the call breakpoint
005064  000753                          BR      5$              ;Br if yes -- Treat like single step break
005066                          6$:     PRINT   <B>
005074  113700  000000G                 MOVB    D.BKNM,R0       ;Get breakpoint number
005100  004737  005772'                 CALL    PRTOCT          ;Print breakpoint number
005104                                  PRINT   <;>
005112  000417                          BR      3$
                                ;
                                ;  See if we had a forced breakpoint
                                ;
005114  032737  000000G 000000G 4$:     BIT     #D$FBRK,D.FLAG  ;Forced breakpoint?
005122  001427                          BEQ     9$              ;Br if not
005124                                  TPRINT  #CRLF           ;Go to a new line
005134                                  PRINT   <Break at >
```

```
                                        ;
                                        ;  Display the address of the breakpoint
                                        ;
005152  013700  0000000         3$:     MOV     D.R7,R0         ;Get address where break occurred
005156  004737  007146'                 CALL    LSTADR          ;Display the address
                                        ;
                                        ;  Display the decoded instruction
                                        ;
005162  013701  0000000                 MOV     D.R7,R1         ;Get address of break
005166  004737  007510'                 CALL    DECODE          ;Display the decoded instruction
005172                                  TPRINT  #CRLF           ;Go to new line
                                        ;
                                        ;  Finished
                                        ;
005202  012601                  9$:     MOV     (SP)+,R1
005204  000207                          RETURN
```

```
                                        .SBTTL  FLGBRK -- Set flag if any breakpoints are active
                                ;-----------------------------------------------------------------------
                                ;  FLGBRK is called to determine if there are any active instruction
                                ;  breakpoints.  If there are, the D$CKBK flag is set in D.FLAG;
                                ;  otherwise the D$CKBK flag is reset.
                                ;
005206  010146                  FLGBRK: MOV     R1,-(SP)
                                ;
                                ;  Determine if there are any normal instruction breakpoints
                                ;
005210  012701  000020                  MOV     #16.,R1         ;Get index to last breakpoint entry
005214  005761  000000          5$:     TST     D.BKAD(R1)      ;Is this breakpoint set?
005220  001007                          BNE     6$              ;Br if yes
005222  162701  000002                  SUB     #2,R1           ;More to check?
005226  002372                          BGE     5$              ;Loop if yes
                                ;
                                ;  No breakpoints are active
                                ;
005230  042737  000000  000000          BIC     #D$CKBK,D.FLAG  ;No breakpoints are set
005236  000403                          BR      9$
                                ;
                                ;  There are active breakpoints
                                ;
005240  052737  000000  000000  6$:     BIS     #D$CKBK,D.FLAG  ;Remember some breakpoint is set
                                ;
                                ;  Finished
                                ;
005246  012601                  9$:     MOV     (SP)+,R1
005250  000207                          RETURN
```

```
                                        .SBTTL  ACRVAL -- Accrue a general value
                                ;-----------------------------------------------------------------------
                                ;  ACRVAL is called to accrue a general value of the form
                                ;  [r,]value    or    $x
                                ;  where "r" is a relocation base register number,
                                ;  and "x" is an internal register number.
                                ;
                                ;  Outputs:
                                ;   R0 = Value accrued.
                                ;   R1 = Status flag:
                                ;        +1 ==> normal value
                                ;         0 ==> No value gotten.
                                ;        -1 ==> Internal register address.
                                ;
005252  010246                  ACRVAL: MOV     R2,-(SP)
                                ;
                                ;  Get 1st character and see if this is a normal value or a register name.
                                ;
005254  004737  005720'                 CALL    GETCHR          ;Get 1st character
005260  120027  000044                  CMPB    R0,#'$          ;Is this a register value?
005264  001461                          BEQ     1$              ;Branch if yes
005266  120027  000056                  CMPB    R0,#'.          ;"." = Current program counter
005272  001451                          BEQ     10$             ;Br if period
005274  004737  005764'                 CALL    PSHCHR          ;Save the character
005300  120027  000053                  CMPB    R0,#'+          ;Is this a sign character?
005304  001414                          BEQ     2$              ;Br if yes
005306  120027  000055                  CMPB    R0,#'-
005312  001411                          BEQ     2$
005314  120027  000060                  CMPB    R0,#'0          ;Is this character a digit?
005320  103403                          BLO     8$              ;Br if not a digit
005322  120027  000071                  CMPB    R0,#'9
005326  101403                          BLOS    2$              ;Br if a digit
                                ;
                                ;  No value was specified
                                ;
005330  005001                  8$:     CLR     R1              ;Return status code
005332  005000                          CLR     R0              ;Return value of 0
005334  000457                          BR      9$
                                ;
                                ;  Accrue normal value
                                ;
005336  004737  005500'          2$:     CALL    ACRNUM          ;Accrue first part of number
                                ;
                                ;  See if 1st part of value is a relocation base register number
                                ;
005342  010102                          MOV     R1,R2           ;Save value that was accrued
005344  120027  000054                  CMPB    R0,#',          ;Was first part terminated with a comma?
005350  001016                          BNE     3$              ;Br if not
005352  020227  000007                  CMP     R2,#7           ;Is first part a valid register number?
005356  101404                          BLOS    4$              ;Br if ok
005360                                  ERR     #EM$IRB
005370  006302                  4$:     ASL     R2              ;Convert reg # to word table index
005372  004737  005720'                 CALL    GETCHR          ;Skip over comma
005376  004737  005500'                 CALL    ACRNUM          ;Accrue second part of number
005402  066201  0000000                 ADD     D.RLBS(R2),R1   ;Add relocation base to offset value
005406  010100                  3$:     MOV     R1,R0           ;Return value in R0
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$:     CMPB    R0,INTCHR(R1)   ;Look up register letter
      71 005444  001406                       BEQ     5$              ;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
```

```
                                        .SBTTL  ACRNUM -- Accrue a number
                                ;-----------------------------------------------------------------
                                ;  ACRNUM is called to accrue a number.
                                ;  The number may be octal or decimal (with a decimal point).
                                ;
                                ;  Outputs:
                                ;   R1 = Value accrued
                                ;   R0 = Delimiter character that was hit.
                                ;
005500  010246                  ACRNUM: MOV     R2,-(SP)
005502  010346                          MOV     R3,-(SP)
005504  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if number has a leading sign character
                                ;
005506  005002                          CLR     R2              ;Assume result should be positive
005510  004737  005720'                 CALL    GETCHR          ;Get 1st character of number
005514  120027  000053                  CMPB    R0,#'+          ;Leading plus sign?
005520  001407                          BEQ     6$              ;Br if yes
005522  120027  000055                  CMPB    R0,#'-          ;Leading minus sign?
005526  001002                          BNE     8$              ;Br if not
005530  005202                          INC     R2              ;Set flag to negate the value
005532  000402                          BR      6$
005534  004737  005764'          8$:     CALL    PSHCHR          ;Push the first digit
                                ;
                                ;  Scan the number and store in D.NMBF
                                ;
005540  012701  0000006         6$:     MOV     #D.NMBF,R1      ;Point to number character buffer
005544  012705  000010                  MOV     #8.,R5          ;Assume this is an octal value
005550  004737  005720'          1$:     CALL    GETCHR          ;Get next character of number
005554  120027  000060                  CMPB    R0,#'0          ;Is this character a digit?
005560  103415                          BLO     2$              ;Br if not
005562  120027  000071                  CMPB    R0,#'9          ;Decimal digit?
005566  101012                          BHI     2$              ;Br if not
005570  120027  000067                  CMPB    R0,#'7          ;Octal digit?
005574  101402                          BLOS    3$              ;Br if yes
005576  012705  000012                  MOV     #10.,R5         ;This must be a decimal value
005602  020127  0000000         3$:     CMP     R1,#D.NMBE      ;Are we about to overflow the buffer?
005606  103040                          BHIS    11$             ;Br if yes
005610  110021                          MOVB    R0,(R1)+        ;Store character into buffer
005612  000756                          BR      1$              ;Loop to get rest of number
                                ;
                                ;  Hit end of number
                                ;
005614  105011                  2$:     CLRB    (R1)            ;Put null at end of buffer
005616  120027  000056                  CMPB    R0,#'.          ;Decimal point at end of number?
005622  001003                          BNE     4$              ;Br if not
005624  012705  000012                  MOV     #10.,R5         ;Remember this is a decimal value
005630  000402                          BR      5$
005632  004737  005764'          4$:     CALL    PSHCHR          ;Save delimiter
                                ;
                                ;  Convert number string to binary value
                                ;
005636  012703  0000006         5$:     MOV     #D.NMBF,R3      ;Point to start of buffer
005642  005001                          CLR     R1              ;Store value in R1
005644  112300                  7$:     MOVB    (R3)+,R0        ;Get next character
005646  001405                          BEQ     9$              ;Br if hit end of number
```

```
      58 005650  162700  000060                    SUB     #'0,R0          ;Convert digit to binary value
      59 005654  070105                            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
```

```
                                        .SBTTL  GETCHR -- Get next character from terminal
                                ;+-----------------------------------------------------------------------
                                ;  Get the next character from the terminal.
                                ;
                                ;  Outputs:
                                ;   R0 = Character gotten
                                ;
005720                          GETCHR:
                                ;
                                ;  See if PSHCHR was called to save a character
                                ;
005720  113700  0000000                 MOVB    D.SVCH,R0       ;Has a character been pushed?
005724  001014                          BNE     9$              ;Br if yes
                                ;
                                ;  No saved character.
                                ;  Get character from terminal.
                                ;
005726                                  .TTYIN                  ;Get character from terminal
005732  005237  0000000                 INC     D.PCOL          ;Advance print column
                                ;
                                ;  Convert lower-case characters to upper-case
                                ;
005736  120027  000141                  CMPB    R0,#141         ;Lower-case a
005742  103405                          BLO     9$
005744  120027  000172                  CMPB    R0,#172         ;Lower-case z
005750  101002                          BHI     9$
005752  162700  000040                  SUB     #40,R0          ;Convert lower-case to upper-case
                                ;
                                ;  Finished
                                ;
005756  105037  0000000         9$:     CLRB    D.SVCH          ;No saved character
005762  000207                          RETURN
                                
                                        .SBTTL  PSHCHR -- Push a character
                                ;+-----------------------------------------------------------------------
                                ;  Push a character so GETCHR will get it on next call.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be pushed.
                                ;
005764  110037  0000000         PSHCHR: MOVB    R0,D.SVCH       ;Save the character for GETCHR
005770  000207                          RETURN                  ;Finished
```

```
                                        .SBTTL  PRTOCT -- Print octal value
                                ;-----------------------------------------------------------------
                                ;  Print an octal value with no leading zeroes.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to print.
                                ;
005772  010146          PRTOCT: MOV     R1,-(SP)
005774  010246                  MOV     R2,-(SP)
005776  010346                  MOV     R3,-(SP)
006000  005003                  CLR     R3              ;Say non-zero digit not seen yet
006002  010001                  MOV     R0,R1           ;Get value to print
006004  005000                  CLR     R0              ;Get 1st bit into R0
006006  073027  000001          ASHC    #1,R0
006012  012702  000006          MOV     #6,R2           ;Print total of 6 digits
006016  000403                  BR      2$
006020  005000          1$:     CLR     R0              ;Get next digit into R0
006022  073027  000003          ASHC    #3,R0
006026  050003          2$:     BIS     R0,R3           ;Remember if non-zero digit seen
006030  001406                  BEQ     3$              ;Br if no non-zero digits seen yet
006032  062700  000060          ADD     #'0,R0          ;Convert to ascii character
006036                          .TTYOUT                 ;Print the character
006042  005237  0000000         INC     D.PCOL          ;Advance print column
006046  077214          3$:     SOB     R2,1$           ;Loop if more digits to print
006050  005703                  TST     R3              ;Did we see any non-zero digits?
006052  001006                  BNE     4$              ;Br if yes
006054                          .TTYOUT #'0             ;Print one zero if not
006064  005237  0000000         INC     D.PCOL          ;Advance print column
006070  012603          4$:     MOV     (SP)+,R3
006072  012602                  MOV     (SP)+,R2
006074  012601                  MOV     (SP)+,R1
006076  000207                  RETURN
```

```
                                        .SBTTL  PRTWRD -- Print octal word value
                                ;-------------------------------------------------------------
                                ;  Print a 16-bit octal value;
                                ;  Inputs:
                                ;   R0 = Value to be printed.
                                ;
006100  010146          PRTWRD: MOV     R1,-(SP)
006102  010246                  MOV     R2,-(SP)
006104  010001                  MOV     R0,R1           ;Get value to print
006106  005000                  CLR     R0              ;Get 1st bit into R0
006110  073027  000001          ASHC    #1,R0
006114  012702  000006          MOV     #6,R2           ;Print total of 6 digits
006120  000403                  BR      2$
006122  005000          1$:     CLR     R0              ;Get next digit into R0
006124  073027  000003          ASHC    #3,R0
006130  062700  000060  2$:     ADD     #'0,R0          ;Convert to ascii character
006134                          .TTYOUT                 ;Print the character
006140  005237  0000000         INC     D.PCOL          ;Advance print column
006144  077212                  SOB     R2,1$           ;Loop if more digits to print
006146  012602                  MOV     (SP)+,R2
006150  012601                  MOV     (SP)+,R1
006152  000207                  RETURN

                                        .SBTTL  PRTBYT -- Print octal byte value
                                ;-------------------------------------------------------------
                                ;  Print an 8-bit octal value.
                                ;
                                ;  Inputs:
                                ;    R0 = Value to be printed.
                                ;
006154  010146          PRTBYT: MOV     R1,-(SP)
006156  010246                  MOV     R2,-(SP)
006160  010001                  MOV     R0,R1           ;Get value to be printed
006162  000301                  SWAB    R1              ;Left-justify value in R1
006164  005000                  CLR     R0              ;Move 1st 2 bits into R0
006166  073027  000002          ASHC    #2,R0
006172  012702  000003          MOV     #3,R2           ;Print 3 digits total
006176  000403                  BR      2$
006200  005000          1$:     CLR     R0              ;Get next octal digit to R0
006202  073027  000003          ASHC    #3,R0
006206  062700  000060  2$:     ADD     #'0,R0          ;Convert to ascii character
006212                          .TTYOUT                 ;Print the character
006216  005237  0000000         INC     D.PCOL          ;Advance print column
006222  077212                  SOB     R2,1$           ;Loop if more digits to print
006224  012602                  MOV     (SP)+,R2
006226  012601                  MOV     (SP)+,R1
006230  000207                  RETURN
```

```
                                        .SBTTL  PRTR50 -- Print a RAD50 value
                                ;-------------------------------------------------------------------
                                ;  PRTR50 is called to print a RAD50 (radix 50) value.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to be printed.
                                ;
006232  010146                  PRTR50: MOV     R1,-(SP)
006234  010246                          MOV     R2,-(SP)
                                ;
                                ;  Convert value to ascii character string and stack the characters
                                ;
006236  012702  000003                  MOV     #3,R2           ;Get # chars to convert
006242  005046                          CLR     -(SP)           ;Put null on stack to signal end
006244  010001                          MOV     R0,R1           ;Get value to be converted
006246  005000                  1$:     CLR     R0              ;Clear high-order for divide
006250  071027  000050                  DIV     #50,R0          ;Divide R0-R1 by 50
006254  116146  000220'                MOVB    R50CHR(R1),-(SP);Convert remainder to ascii and stack it
006260  010001                          MOV     R0,R1           ;Get quotient
006262  077207                          SOB     R2,1$           ;Loop if more to convert
                                ;
                                ;  Finished conversion.  Print the result.
                                ;
006264  012600                  2$:     MOV     (SP)+,R0        ;Get char to print
006266  001405                          BEQ     3$              ;Br if hit end of string
006270                                  .TTYOUT                 ;Print the character
006274  005237  0000000                 INC     @#D.PCOL        ;Advance print column
006300  000771                          BR      2$              ;Loop to print more
                                ;
                                ;  Finished
                                ;
006302  012602                  3$:     MOV     (SP)+,R2
006304  012601                          MOV     (SP)+,R1
006306  000207                          RETURN
```

```
 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  000000G                        CLR     D.PCOL          ;Say print column = 0
 9 006332  013706  000000G                        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  000000G                        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
```

```
      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,R0       ;Get mode of address
     14 006374  001405                           BEQ     9$              ;Br if no address specified
     15 006376  110037  0000000                  MOVB    R0,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
```

```
                                        .SBTTL  STRVAL -- Store value into open cell
                                ;-----------------------------------------------------------------
                                ;  STRVAL is called to store a value into the currently open cell.
                                ;
                                ;  Inputs:
                                ;   R0      = Value to store.
                                ;   D.LOC   = Address where to store value.
                                ;   D.LOCM  = +1=Store into user's area, -1=Store into internal register.
                                ;   D.BYTM  = 0=Word mode store, 1=Byte mode store.
                                ;
006412  010146                  STRVAL: MOV     R1,-(SP)
006414  010246                          MOV     R2,-(SP)
                                ;
                                ;  Make sure a cell is open
                                ;
006416  105737  0000000                 TSTB    D.LOCM          ;Is a cell open now?
006422  001471                          BEQ     9$              ;Br if not
                                ;
                                ;  Determine if we are storing into user's job or internal cell
                                ;
006424  105737  0000000                 TSTB    D.LOCM          ;Internal or user job?
006430  100436                          BMI     1$              ;Br if internal cell
                                ;
                                ;  Store into user's job
                                ;
006432  105737  0000000                 TSTB    D.BYTM          ;Byte or word mode?
006436  001005                          BNE     2$              ;Br if byte mode
                                ;
                                ;  Store into word
                                ;
006440  010046                          MOV     R0,-(SP)        ;Put value on stack
006442  013700  0000000                 MOV     D.LOC,R0        ;Get address where we should store
006446  106610                          MTPD    (R0)            ;Store into user's area
006450  000456                          BR      9$
                                ;
                                ;  Store into byte
                                ;
006452  042700  177400          2$:     BIC     #^C377,R0       ;Clear high-order byte of value being stored
006456  013702  0000000                 MOV     D.LOC,R2        ;Get address where we are to store
006462  042702  000001                  BIC     #1,R2           ;Force address to be even
006466  106512                          MFPD    (R2)            ;Get current contents of word
006470  032737  000001  0000000         BIT     #1,D.LOC        ;Even or odd byte?
006476  001005                          BNE     3$              ;Br if odd byte
006500  042716  000377                  BIC     #377,(SP)       ;Clear low-order byte of old value
006504  050016                          BIS     R0,(SP)         ;Put in new low-order byte
006506  106612                          MTPD    (R2)            ;Store new word
006510  000436                          BR      9$
006512  042716  177400          3$:     BIC     #177400,(SP)    ;Clear high-order byte
006516  000300                          SWAB    R0              ;Get new value to high-order byte
006520  050016                          BIS     R0,(SP)         ;Store new high-order byte
006522  106612                          MTPD    (R2)            ;Store new word
006524  000430                          BR      9$
                                ;
                                ;  Store into internal register
                                ;
006526  105737  0000000         1$:     TSTB    D.BYTM          ;Byte or word mode?
006532  001003                          BNE     12$             ;Br if byte mode
```

```
                                        ;
                                        ;  Store into word
                                        ;
006534  010077  0000000                         MOV     R0,@D.LOC       ;Store into internal cell
006540  000422                                  BR      9$
                                        ;
                                        ;  Store into byte
                                        ;
006542  042700  177400          12$:    BIC     #^C377,R0       ;Clear high-order byte of value being stored
006546  013702  0000000                 MOV     D.LOC,R2        ;Get address where we are to store
006552  042702  000001                  BIC     #1,R2           ;Force address to be even
006556  032737  000001  0000000         BIT     #1,D.LOC        ;Even or odd byte?
006564  001352                          BNE     3$              ;Br if odd byte
006566  042712  000377                  BIC     #377,(R2)       ;Clear low-order byte of old value
006572  050012                          BIS     R0,(R2)         ;Put in new low-order byte
006574  000404                          BR      9$
006576  042712  177400          13$:    BIC     #177400,(R2)    ;Clear high-order byte
006602  000300                          SWAB    R0              ;Get new value to high-order byte
006604  050012                          BIS     R0,(R2)         ;Store new high-order byte
                                        ;
                                        ;  Finished
                                        ;
006606  012602                  9$:     MOV     (SP)+,R2
006610  012601                          MOV     (SP)+,R1
006612  000207                          RETURN
```

```
                                        .SBTTL  GETVAL -- Get value from current cell
                                ;-----------------------------------------------------------------
                                ;   GETVAL is called to get the value from the currently open cell.
                                ;
                                ;  Inputs:
                                ;   D.LOC  = Address of cell from which value is to be fetched.
                                ;   D.LOCM = +1=fetch from user's area, -1=Fetch from internal register.
                                ;   D.BYTM = 0=Word mode, 1=Byte mode
                                ;
                                ;  Outputs:
                                ;   R0 = Value obtained
                                ;
006614  010246                  GETVAL: MOV     R2,-(SP)
                                ;
                                ;  Determine if we are fetching from user's job or internal cell
                                ;
006616  105737  0000000                 TSTB    D.LOCM          ;Internal or user job?
006622  100426                          BMI     1$              ;Br if internal cell
                                ;
                                ;  Fetch from user's job
                                ;
006624  105737  0000000                 TSTB    D.BYTM          ;Byte or word mode?
006630  001005                          BNE     2$              ;Br if byte mode
                                ;
                                ;  Fetch from word
                                ;
006632  013700  0000000                 MOV     D.LOC,R0        ;Get address of value wanted
006636  106510                          MFPD    (R0)            ;Get value from user's area
006640  012600                          MOV     (SP)+,R0        ;Return in R0
006642  000440                          BR      9$
                                ;
                                ;  Fetch from byte
                                ;
006644  013702  0000000         2$:     MOV     D.LOC,R2        ;Get address of word
006650  042702  000001                  BIC     #1,R2           ;Force address to be even
006654  106512                          MFPD    (R2)            ;Get current contents of word
006656  012600                          MOV     (SP)+,R0        ;Get word value to R0
006660  032737  000001  0000000         BIT     #1,D.LOC        ;Even or odd byte?
006666  001401                          BEQ     3$              ;Br if low-order byte wanted
006670  000300                          SWAB    R0              ;Swap bytes if high-order byte wanted
006672  042700  177400          3$:     BIC     #177400,R0      ;Clear high-order byte
006676  000422                          BR      9$
                                ;
                                ;  Fetch from internal register
                                ;
006700  105737  0000000         1$:     TSTB    D.BYTM          ;Byte or word mode?
006704  001003                          BNE     12$             ;Br if byte mode
                                ;
                                ;  Fetch from word
                                ;
006706  017700  0000000                 MOV     @D.LOC,R0       ;Get the value
006712  000414                          BR      9$
                                ;
                                ;  Fetch from byte
                                ;
006714  013702  0000000         12$:    MOV     D.LOC,R2        ;Get address where we are to fetch
006720  042702  000001                  BIC     #1,R2           ;Force address to be even
```

```
     58 006724  011200                          MOV     (R2),R0         ;Get the word value
     59 006726  032737  000001  000006          BIT     #1,D.LOC        ;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
```

```
                                        .SBTTL  SHOLOC -- Display current address
                                ;-----------------------------------------------------------------
                                ;  Display address of currently open cell.
                                ;  The address may be either in the user's program space
                                ;  or may be the address of an internal register ($x).
                                ;
                                ;  Inputs:
                                ;   D.LOC  = Address to be displayed
                                ;   D.LOCM = Mode of address (+1==>In user's program, -1==>Internal cell)
                                ;
006750  010146                  SHOLOC: MOV     R1,-(SP)
006752  010246                          MOV     R2,-(SP)
006754  010346                          MOV     R3,-(SP)
006756  010446                          MOV     R4,-(SP)
                                ;
                                ;  Determine if this is an internal or external cell
                                ;
006760  105737  000000G                 TSTB    D.LOCM          ;Internal or external address?
006764  002405                          BLT     1$              ;Br if internal address
                                ;
                                ;  Display address of cell in user's program
                                ;
006766  013700  000000G                 MOV     D.LOC,R0        ;Get address to be displayed
006772  004737  007146'                 CALL    LSTADR          ;Display relative to relocation base
006776  000456                          BR      9$
                                ;
                                ;  Display name of internal cell
                                ;
007000  013700  000000G         1$:     MOV     D.LOC,R0        ;Get address of internal cell
007004  012701  000400'                 MOV     #INTADR,R1      ;Point to table of addresses of cells
007010  012702  177777                  MOV     #-1,R2          ;Init for search
007014  005003                          CLR     R3
007016  020011                  2$:     CMP     R0,(R1)         ;Is address below base of this cell?
007020  103406                          BLO     3$              ;Br if yes
007022  010004                          MOV     R0,R4           ;Get address of interest
007024  161104                          SUB     (R1),R4         ;Determine distance from cell base addr
007026  020402                          CMP     R4,R2           ;Is this the closest we've seen so far?
007030  101002                          BHI     3$              ;Br if not
007032  010402                          MOV     R4,R2           ;Remember offset from nearest cell
007034  010103                          MOV     R1,R3           ;Remember address of nearest cell
007036  062701  000002          3$:     ADD     #2,R1           ;Point to address of next cell
007042  020127  000432'                 CMP     R1,#INTEND      ;Checked all cells?
007046  103763                          BLO     2$              ;Loop if not
                                ;
                                ;  Display register name
                                ;
007050  162703  000400'                 SUB     #INTADR,R3      ;Get offset into address table
007054  006203                          ASR     R3              ;Convert to byte table index
007056                                  .TTYOUT #'$             ;Print "$"
007066  005237  000000G                 INC     D.PCOL          ;Advance print column
007072  116300  000362'                 MOVB    INTCHR(R3),R0   ;Get name of internal register
007076                                  .TTYOUT                 ;Print register name
007102  005237  000000G                 INC     D.PCOL          ;Advance print column
                                ;
                                ;  Display offset from register
                                ;
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
```

```
                                        .SBTTL  LSTADR -- Display address in user's program
                                ;-----------------------------------------------------------------------
                                ;  LSTADR is called to display an address in the user's program.
                                ;  LSTADR functions the same as SHOADR except that LSTADR checks the
                                ;  DP$LAA flag in the $F format register.  If the DP$LAA flag is cleared
                                ;  the address is displayed in relative form; if it is set, it is displayed
                                ;  in absolute form.
                                ;
                                ;  Inputs:
                                ;   R0 = Address to be displayed.
                                ;
007146                          LSTADR:
                                ;
                                ;  See if DP$LAA is set in $F register
                                ;
007146  032737  0000000G 0000000G       BIT     #DP$LAA,D.PFMT  ;Absolute addresses wanted?
007154  001003                          BNE     1$              ;Br if yes
                                ;
                                ;  Display address with relocation base
                                ;
007156  004737  007172'                 CALL    SHOADR          ;Display address with relocation base
007162  000402                          BR      9$
                                ;
                                ;  Display absolute address
                                ;
007164  004737  005772'         1$:     CALL    PRTOCT          ;Display absolute address
                                ;
                                ;  Finished
                                ;
007170  000207                  9$:     RETURN
```

```
                                        .SBTTL  SHOADR -- Display address in user's program
                                ;-----------------------------------------------------------------
                                ;  SHOADR is called to display an address in the user's program.
                                ;  A check is made to determine which relocation register the address
                                ;  is relative to and if any, the address is printed as "r,o".
                                ;
                                ;  Inputs:
                                ;   R0 = Address to display.
                                ;
007172  010146                  SHOADR: MOV     R1,-(SP)
007174  010246                          MOV     R2,-(SP)
007176  010346                          MOV     R3,-(SP)
007200  010446                          MOV     R4,-(SP)
                                ;
                                ;  Determine if address is relative to a relocation register
                                ;
007202  010004                          MOV     R0,R4           ;Save original address
007204  004737  007262'                  CALL    RELADR          ;Determine if addr is relative to reloc regn
007210  103414                          BCS     4$              ;Br if not
                                ;
                                ;  Print relocation register number
                                ;
007212  166304  0000000                 SUB     D.RLBS(R3),R4   ;Compute offset within the region
007216  010300                          MOV     R3,R0           ;Get register index
007220  006200                          ASR     R0              ;Convert index to number
007222  004737  005772'                  CALL    PRTOCT          ;Print register number
007226                                  .TTYOUT #54             ;Print ","
007236  005237  0000000                 INC     D.PCOL          ;Advance print column
                                ;
                                ;  Print offset within region
                                ;
007242  010400                  4$:     MOV     R4,R0           ;Get offset within region
007244  004737  005772'                  CALL    PRTOCT          ;Print the value
                                ;
                                ;  Finished
                                ;
007250  012604                          MOV     (SP)+,R4
007252  012603                          MOV     (SP)+,R3
007254  012602                          MOV     (SP)+,R2
007256  012601                          MOV     (SP)+,R1
007260  000207                          RETURN
```

```
                                        .SBTTL  RELADR -- Determine if address is in relocation region
                                ;-------------------------------------------------------------------
                                ;  RELADR is called to determine if an address is within a relocation
                                ;  region.
                                ;
                                ;  Inputs:
                                ;   R0 = Address to be checked.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Address is within a relocation region.
                                ;   C-flag set     ==> Address is not within a relocation region.
                                ;   R3 = Index to relocation region to be used with address.
                                ;
007262  010146          RELADR: MOV     R1,-(SP)
007264  010246                  MOV     R2,-(SP)
007266  010446                  MOV     R4,-(SP)
                                ;
                                ;  Search for relocation region defined below the address
                                ;
007270  005001                  CLR     R1              ;Init relocation register index
007272  012702  177777          MOV     #-1,R2          ;Init closest offset value
007276  010203                  MOV     R2,R3           ;Say no relocation base found yet
007300  005761  000000  2$:     TST     D.RLBS(R1)      ;Is this relocation register in use?
007304  001412                  BEQ     3$              ;Br if not
007306  020061  000000          CMP     R0,D.RLBS(R1)   ;Is address above this relocation base?
007312  103407                  BLO     3$              ;Br if not
007314  010004                  MOV     R0,R4           ;Get address of interest
007316  166104  000000          SUB     D.RLBS(R1),R4   ;Calculate offset within region
007322  020402                  CMP     R4,R2           ;Is this the closest one so far?
007324  101002                  BHI     3$              ;Br if not
007326  010402                  MOV     R4,R2           ;Rember smallest offset
007330  010103                  MOV     R1,R3           ;Remember relocation register index
007332  062701  000002  3$:     ADD     #2,R1           ;Advance relocation register index
007336  020127  000016          CMP     R1,#14.         ;Checked all relocation registers?
007342  101756                  BLOS    2$              ;Br if not
                                ;
                                ;  See if we found a relocation region
                                ;
007344  005703                  TST     R3              ;Did we find a relocation region
007346  002002                  BGE     5$              ;Br if yes
007350  000261                  SEC                     ;Say no region found
007352  000401                  BR      9$
007354  000241          5$:     CLC                     ;Say a region was found
                                ;
                                ;  Finished
                                ;
007356  012604          9$:     MOV     (SP)+,R4
007360  012602                  MOV     (SP)+,R2
007362  012601                  MOV     (SP)+,R1
007364  000207                  RETURN
```

```
                                        .SBTTL  DOPRT  -- Print a text string
                                ;-----------------------------------------------------------------
                                ;  DOPRT is called to print a text string.
                                ;  It directly calls routines in TSTTY to print each character.
                                ;
                                ;  Inputs:
                                ;   R0 = Pointer to asciz string.
                                ;
007366  010146                  DOPRT:  MOV     R1,-(SP)
007370  010246                          MOV     R2,-(SP)
                                ;
                                ;  Set up pointer to start of string
                                ;
007372  010002                          MOV     R0,R2           ;Get pointer to start of string
007374  113701  0000000                 MOVB    @#RUSR,R1       ;Get current job index number
                                ;
                                ;  Get each char out of the string and print it
                                ;
007400  112200                  1$:     MOVB    (R2)+,R0        ;Get next char from string
007402  001411                          BEQ     2$              ;Br if hit null at end
007404  120027  000200                  CMPB    R0,#200         ;Is string terminated with a 200?
007410  001422                          BEQ     9$              ;Br if yes
007412                                  @CALL   PUTCHR          ;Print the character
007420  005237  0000000                 INC     @#D.PCOL        ;Advance print column
007424  000765                          BR      1$              ;Loop to print rest of string
                                ;
                                ;  Print Carriage-return Line-feed at end of string
                                ;
007426  012700  000015          2$:     MOV     #CR,R0          ;Get carriage return
007432                                  @CALL   PUTCHR          ;Print it
007440  012700  000012                  MOV     #LF,R0          ;Get line feed
007444                                  @CALL   PUTCHR          ;Print it
007452  005037  0000000                 CLR     @#D.PCOL        ;Say we are back to column 1
                                ;
                                ;  Finished
                                ;
007456  012602                  9$:     MOV     (SP)+,R2
007460  012601                          MOV     (SP)+,R1
007462  000207                          RETURN
```

```
                                        .SBTTL  DBGON  -- Place terminal in debug mode
                                ;-----------------------------------------------------------------
                                ;  DBGON is called to place the terminal in debug mode.
                                ;
007464  012700  000002'         DBGON:  MOV     #ONEMT,R0
007470  104375                         EMT     375
007472  000207                         RETURN
                                
                                        .SBTTL  DBGOFF -- Turn off terminal debug mode
                                ;-----------------------------------------------------------------
                                ;  DBGOFF is called to take the terminal out of debug mode
                                ;
007474  012700  000004'         DBGOFF: MOV     #OFFEMT,R0
007500  104375                         EMT     375
007502  000207                         RETURN
                                
                                        .SBTTL  CHKADR -- Determine if address is valid
                                ;-----------------------------------------------------------------
                                ;  CHKADR is called to determine if an address is within the
                                ;  range of the user's program space.
                                ;
                                ;  Inputs:
                                ;   R0 = Address to be checked.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> address is ok
                                ;   C-flag set     ==> address is invalid
                                ;
007504  000241                  CHKADR: CLC
007506  000207                         RETURN
```

```
                                        .SBTTL  DECODE -- Decode instruction into symbolic form
                                ;-----------------------------------------------------------------------
                                ;  DECODE is called to decode an instruction word into its symbolic form.
                                ;  The symbolic form of the instruction is displayed on the console.
                                ;
                                ;  Inputs:
                                ;   R1 = Address of instruction to be decoded.
                                ;
                                ;  Outputs:
                                ;   D.ILEN = Length (in bytes) of decoded instruction and operands.
                                ;
007510  010146                  DECODE: MOV     R1,-(SP)
007512  010246                          MOV     R2,-(SP)
007514  010346                          MOV     R3,-(SP)
007516  010446                          MOV     R4,-(SP)
007520  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize instruction length to 2 bytes
                                ;
007522  012737  000002  000000G         MOV     #2,D.ILEN       ;Say base instruction length = 2 bytes
                                ;
                                ;  Initialize column counter for decoded instruction display
                                ;
007530                                  PRINT   <    >          ;First print a few spaces
007542  005037  0000000                 CLR     D.PCOL          ;Say we are at column 1 now
                                ;
                                ;  Fetch the instruction word
                                ;
007546  106521                          MFPD    (R1)+           ;Get the instruction from user's space
007550  012602                          MOV     (SP)+,R2        ;Get instruction off of stack
                                ;
                                ;  Search the instruction list for this instruction
                                ;
007552  012704  011764'                 MOV     #INSBAS,R4      ;Point to base of instruction decode table
007556  010203                  1$:     MOV     R2,R3           ;Get instruction to be decoded
007560  046403  000000                  BIC     IB$MSK(R4),R3   ;Clear operand fields in instruction
007564  020364  000002                  CMP     R3,IB$VAL(R4)   ;Is this the instruction?
007570  001411                          BEQ     5$              ;Br if found the instruction
007572  116400  000005                  MOVB    IB$LEN(R4),R0   ;Get length of ascii name string
007576  005200                          INC     R0              ;Bound up to word boundary
007600  042700  000001                  BIC     #1,R0
007604  062700  000006                  ADD     #IB$NAM,R0      ;Add size of base portion of block
007610  060004                          ADD     R0,R4           ;Point to next instruction decode block
007612  000761                          BR      1$              ;Continue searching for the instruction
                                ;
                                ;  We have found the instruction in the instruction decode table.
                                ;  R4 = Address of instruction decode block.
                                ;  Display the symbolic name of the instruction.
                                ;
007614  116403  000005          5$:     MOVB    IB$LEN(R4),R3   ;Get length of instruction mnemonic
007620  060337  0000000                 ADD     R3,D.PCOL       ;Advance column counter
007624  010405                          MOV     R4,R5           ;Point to the mnemonic ascii string
007626  062705  000006                  ADD     #IB$NAM,R5
007632  112500                  6$:     MOVB    (R5)+,R0        ;Get next char of mnemonic
007634                                  .TTYOUT                 ;Print the character
007640  077304                          SOB     R3,6$           ;Loop if more chars to print
                                ;
```

```
                                        ;  Now mask out the instruction code from the instruction word leaving
                                        ;  only the operand fields.
                                        ;
007642  016400  000000                        MOV     IB$MSK(R4),R0   ;Get mask value
007646  005100                                COM     R0              ;Complement it to leave the operand fields
007650  040002                                BIC     R0,R2           ;Mask out all but the operand fields
                                        ;
                                        ;  Now call appropriate routine to display the operand values for
                                        ;  this instruction.
                                        ;   R1 = Address of word past instruction.
                                        ;   R2 = Operand field values (instruction word with instruction code masked)
                                        ;
007652  116400  000004                        MOVB    IB$TYP(R4),R0   ;Get the operand type code for this inst
007656  004770  014200'                       CALL    @TYPVEC(R0)     ;Call appropriate operand display routine
                                        ;
                                        ;  Finished
                                        ;
007662  012605                                MOV     (SP)+,R5
007664  012604                                MOV     (SP)+,R4
007666  012603                                MOV     (SP)+,R3
007670  012602                                MOV     (SP)+,R2
007672  012601                                MOV     (SP)+,R1
007674  000207                                RETURN
```

```
                                        .SBTTL  ODCxxx -- Display instruction operands
                                ;-----------------------------------------------------------------------
                                ;  TYP1 -- Instruction has no operands.
                                ;
007676  000207                  ODC1:   RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP2 -- Operand has a single general operand.
                                ;
007700                          ODC2:
                                ;
                                ;  Tab over to operand field
                                ;
007700  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
007704  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the operand
                                ;
007710  010200                          MOV     R2,R0           ;Get the operand code
007712  004737  010650'                 CALL    ODCGEN          ;Display general operand
                                ;
                                ;  Finished
                                ;
007716  000207                          RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP3 -- Single register operand.
                                ;
007720                          ODC3:
                                ;
                                ;  Tab over to the operand field
                                ;
007720  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
007724  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the register
                                ;
007730  010200                          MOV     R2,R0           ;Get the register number
007732  004737  011342'                 CALL    ODCREG          ;Display the register number
007736  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP4 -- 4-bit absolute numeric operand
                                ;
007740                          ODC4:
                                ;
                                ;  Tab over to the operand field
                                ;
007740  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
007744  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the value
                                ;
007750  010200                          MOV     R2,R0           ;Get value to display
007752  004737  005772'                 CALL    PRTOCT          ;Print the value
                                ;
                                ;  Finished
                                ;
007756  000207                          RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP5 -- Condition code modification instruction
                                ;
007760                          ODC5:
                                ;
                                ;  Determine which condition code is affected
                                ;
007760  032702  000001                  BIT     #1,R2           ;C-flag?
007764  001404                          BEQ     1$              ;Br if not
007766                                  .TTYOUT #'C
007776  032702  000002          1$:     BIT     #2,R2           ;V-flag?
010002  001404                          BEQ     2$              ;Br if not
010004                                  .TTYOUT #'V
010014  032702  000004          2$:     BIT     #4,R2           ;Z-flag?
010020  001404                          BEQ     3$              ;Br if not
010022                                  .TTYOUT #'Z
010032  032702  000010          3$:     BIT     #10,R2          ;N-flag?
010036  001404                          BEQ     4$              ;Br if not
010040                                  .TTYOUT #'N
                                ;
                                ;  Finished
                                ;
010050  005237  000000G         4$:     INC     D.PCOL          ;Advance column number
010054  000207                          RETURN
```

```
                                ;+-----------------------------------------------------------------------------
                                ;  TYP6 -- Branch offset operand
                                ;
010056                          ODC6:
                                ;
                                ;  Tab over to the operand field
                                ;
010056  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010062  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the branch target
                                ;
010066  010200                          MOV     R2,R0           ;Get instruction word
010070  042700  177400                  BIC     #^C377,R0       ;Leave only branch offset
010074  032700  000200                  BIT     #200,R0         ;Is offset negative?
010100  001402                          BEQ     1$              ;Br if not
010102  052700  177400                  BIS     #177400,R0      ;Sign extend the offset
010106  006300                  1$:     ASL     R0              ;Convert word offset to byte offset
010110  060100                          ADD     R1,R0           ;Add current PC address
010112  004737  011570'                 CALL    ODCADR          ;Display the address
                                ;
                                ;  Finished
                                ;
010116  000207                          RETURN

                                ;+-----------------------------------------------------------------------------
                                ;  TYP7 -- Register and general destination.
                                ;
010120                          ODC7:
                                ;
                                ;  Tab over to the operand field
                                ;
010120  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010124  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the register
                                ;
010130  010200                          MOV     R2,R0           ;Get instruction word
010132  072027  177772                  ASH     #-6,R0          ;Right justify the register value
010136  004737  011342'                 CALL    ODCREG          ;Display the register name
010142                                  .TTYOUT #COMMA          ;Display ","
010152  005237  000000G                 INC     D.PCOL          ;Inc column counter
                                ;
                                ;  Display the destination address
                                ;
010156  010200                          MOV     R2,R0           ;Get the destination address descriptor
010160  004737  010650'                 CALL    ODCGEN          ;Display the address
                                ;
                                ;  Finished
                                ;
010164  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP8 -- Register destination and general source operand (SS,R)
                                ;
010166                          ODC8:
                                ;
                                ;  Tab over to the operand field
                                ;
010166  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010172  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the source item
                                ;
010176  010200                          MOV     R2,R0           ;Get the source operand code
010200  004737  010650'                 CALL    ODCGEN          ;Display it
010204                                  .TTYOUT #COMMA          ;Display ","
010214  005237  000000                  INC     @.PCOL          ;Advance column counter
                                ;
                                ;  Display the register destination
                                ;
010220  010200                          MOV     R2,R0           ;Get the register number
010222  072027  177772                  ASH     #-6,R0          ;Right justify
010226  004737  011342'                 CALL    ODCREG          ;Display the register
                                ;
                                ;  Finished
                                ;
010232  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP9 -- Two general operands
                                ;
010234                          ODC9:
                                ;
                                ;  Tab over to the operand field
                                ;
010234  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010240  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the source operand
                                ;
010244  010200                          MOV     R2,R0           ;Get instruction word
010246  072027  177772                  ASH     #-6,R0          ;Right justify the source item
010252  004737  010650'                 CALL    ODCGEN          ;Display source item
010256                                  .TTYOUT #COMMA          ;Print a comma
                                ;
                                ;  Display the destination operand
                                ;
010266  010200                          MOV     R2,R0           ;Get destination item code
010270  004737  010650'                 CALL    ODCGEN          ;Display destination item
                                ;
                                ;  Finished
                                ;
010274  000207                          RETURN
```

```
                                ;+-----------------------------------------------------------------------------
                                ;  TYP10 -- SOB instruction
                                ;
010276                          ODC10:
                                ;
                                ;  Tab over to the operand field
                                ;
010276  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010302  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the register number
                                ;
010306  010200                          MOV     R2,R0           ;Get instruction word
010310  072027  177772                  ASH     #-6,R0          ;Right justify register number
010314  004737  011342'                 CALL    ODCREG          ;Display the register name
010320                                  .TTYOUT #COMMA          ;Print comma
010330  005237  000000G                 INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the destination address
                                ;
010334  042702  177700                  BIC     #^C77,R2        ;Mask out all but branch offset
010340  010100                          MOV     R1,R0           ;Get the current PC address
010342  006302                          ASL     R2              ;Get 2* offset
010344  160200                          SUB     R2,R0           ;Get destination address
010346  004737  011570'                 CALL    ODCADR          ;Display the address
                                ;
                                ;  Finished
                                ;
010352  000207                  RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP11 -- EMT and TRAP instructions
                                ;
010354                          ODC11:
                                ;
                                ;  Tab over to the operand field
                                ;
010354  012700  000010                MOV     #COLOPN,R0      ;Get operand column number
010360  004737  011734'               CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the argument value
                                ;
010364  010200                        MOV     R2,R0           ;Get instruction word
010366  042700  177400                BIC     #^C377,R0       ;Mask out all but operand value
010372  004737  005772'               CALL    PRTOCT          ;Print octal value
                                ;
                                ;  Finished
                                ;
010376  000207                        RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP12 -- Single floating-point operand
                                ;
010400                          ODC12:
                                ;
                                ;  Tab over to the operand field
                                ;
010400  012700  000010                MOV     #COLOPN,R0      ;Get operand column number
010404  004737  011734'               CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the floating point operand
                                ;
010410  010200                        MOV     R2,R0           ;Get operand code
010412  004737  011542'               CALL    ODCFPU          ;Display general FPU operand
                                ;
                                ;  Finished
                                ;
010416  000207                        RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP13 -- Floating accumulator and floating source operand.
                                ;
010420                          ODC13:
                                ;
                                ;  Tab over to the operand field
                                ;
010420  012700  000010                MOV     #COLOPN,R0      ;Get operand column number
010424  004737  011734'               CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the source operand
                                ;
010430  010200                        MOV     R2,R0           ;Get instruction word
010432  004737  011542'               CALL    ODCFPU          ;Display the source operand
010436                                .TTYOUT #COMMA          ;Display comma
010446  005237  0000000               INC     D.PCOL          ;Advance column number
                                ;
                                ;  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
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP14 -- Floating accumulator and general source operand.
                                ;
 010466                         ODC14:
                                ;
                                ;  Tab over to the operand field
                                ;
 010466  012700  000010                 MOV     #COLOPN,R0      ;Get operand column number
 010472  004737  011734'                CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the source operand
                                ;
 010476  010200                         MOV     R2,R0           ;Get instruction word
 010500  004737  010650'                CALL    ODCGEN          ;Display the source operand
 010504                                 .TTYOUT #COMMA          ;Display comma
 010514  005237  000000G                INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the accumulator number
                                ;
 010520  010200                         MOV     R2,R0           ;Get accumulator number
 010522  072027  177772                 ASH     #-6,R0          ;Right justify
 010526  004737  011470'                CALL    ODCACC          ;Display the accumulator number
                                ;
                                ;  Finished
                                ;
 010532  000207                         RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP15 -- Floating accumulator and floating destination operand.
                                ;
010534                          ODC15:
                                ;
                                ;  Tab over to the operand field
                                ;
010534  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010540  004737  011734'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the floating accumulator
                                ;
010544  010200                          MOV     R2,R0           ;Get the accumulator number
010546  072027  177772                  ASH     #-6,R0          ;Right justify
010552  004737  011470'                 CALL    ODCACC          ;Display the accumulator
010556                                  .TTYOUT #COMMA          ;Display comma
010566  005237  0000000                 INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the floating destination
                                ;
010572  010200                          MOV     R2,R0           ;Get the destination operand code
010574  004737  011542'                 CALL    ODCFPU          ;Display it
                                ;
                                ;  Finished
                                ;
010600  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP16 -- Floating accumulator and general destination operand.
                                ;
010602                          ODC16:
                                ;
                                ;  Tab over to the operand field
                                ;
010602  012700  000010                MOV     #COLOPN,R0      ;Get operand column number
010606  004737  011734'               CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the floating accumulator
                                ;
010612  010200                        MOV     R2,R0           ;Get the accumulator number
010614  072027  177772                ASH     #-6,R0          ;Right justify
010620  004737  011470'               CALL    ODCACC          ;Display the accumulator
010624                                .TTYOUT #COMMA          ;Display comma
010634  005237  0000000               INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the destination
                                ;
010640  010200                        MOV     R2,R0           ;Get the destination operand code
010642  004737  010650'               CALL    ODCGEN          ;Display it
                                ;
                                ;  Finished
                                ;
010646  000207                        RETURN
```

```
                                        .SBTTL  ODCGEN -- Display general operand value
                                ;-----------------------------------------------------------------
                                ;  ODCGEN is called during instruction decoding to display a general
                                ;  operand of any mode.
                                ;
                                ;  Inputs:
                                ;   R0 = Operand value from instruction.
                                ;   R1 = Address of word following instruction.
                                ;
                                ;  Outputs:
                                ;   R1 is incremented if the addressing mode uses a value in memory that
                                ;   follows the instruction.
                                ;
010650  010246                  ODCGEN: MOV     R2,-(SP)
010652  010346                          MOV     R3,-(SP)
                                ;
                                ;  Get the addressing mode
                                ;
010654  010002                          MOV     R0,R2
010656  072227  177776                  ASH     #-2,R2          ;Right justify 2* mode
010662  042702  177761                  BIC     #^C16,R2        ;Mask out all but 2* mode
                                ;
                                ;  Determine if the register is R7 or is in the range R0 to R6.
                                ;
010666  010003                          MOV     R0,R3           ;Get mode and register value
010670  042703  177770                  BIC     #^C7,R3         ;Mask out all but register number
010674  020327  000007                  CMP     R3,#7           ;Is register R7?
010700  001003                          BNE     1$
                                ;
                                ;  Addressing mode is relative to R7.
                                ;  Call display routine based on the addressing mode.
                                ;   R0 and R3 contain the mode and register number.
                                ;
010702  004772  011322'                 CALL    @AD7VEC(R2)     ;Call the processing routine
010706  000402                          BR      9$
                                ;
                                ;  Jump to processing routine based on the mode
                                ;
010710  004772  011302'         1$:     CALL    @ADRVEC(R2)     ;Jump to processing routine
                                ;
                                ;  Finished
                                ;
010714  012603                  9$:     MOV     (SP)+,R3
010716  012602                          MOV     (SP)+,R2
010720  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Display routines for registers R0 through R6
                                ;
                                ;  Mode 0 -- R
                                ;
010722  004737  011342'         ADRMD0: CALL    ODCREG          ;Display register
010726  000207                          RETURN
                                ;
                                ;  Mode 1 -- (R)
                                ;
010730  010046                  ADRMD1: MOV     R0,-(SP)        ;Save the register number
010732                                  .TTYOUT #'(             ;Open paren
010742  012600                          MOV     (SP)+,R0        ;Recover the register number
010744  004737  011342'                 CALL    ODCREG          ;Display the register number
010750                                  .TTYOUT #')             ;Close paren
010760  000207                          RETURN
                                ;
                                ;  Mode 2 -- (R)+
                                ;
010762  004737  010730'         ADRMD2: CALL    ADRMD1          ;Print "(r)"
010766                                  .TTYOUT #'+             ;Put in trailing plus sign
010776  000207                          RETURN
                                ;
                                ;  Mode 3 -- @(R)+
                                ;
011000  010046                  ADRMD3: MOV     R0,-(SP)        ;Save the register number
011002                                  .TTYOUT #'@             ;Print "@"
011012  012600                          MOV     (SP)+,R0        ;Recover the register number
011014  004737  010762'                 CALL    ADRMD2          ;Print "(r)+"
011020  000207                          RETURN
                                ;
                                ;  Mode 4 --  -(R)
                                ;
011022  010046                  ADRMD4: MOV     R0,-(SP)        ;Save the register number
011024                                  .TTYOUT #'-             ;Print "-"
011034  012600                          MOV     (SP)+,R0        ;Recover the register number
011036  004737  010730'                 CALL    ADRMD1          ;Print "(r)"
011042  000207                          RETURN
                                ;
                                ;  Mode 5 -- @-(R)
                                ;
011044  010046                  ADRMD5: MOV     R0,-(SP)        ;Save the register number
011046                                  .TTYOUT #'@             ;Print "@-"
011056                                  .TTYOUT #'-
011066  012600                          MOV     (SP)+,R0        ;Recover register number
011070  004737  010730'                 CALL    ADRMD1          ;Print "(r)"
011074  000207                          RETURN
                                ;
                                ;  Mode 6 -- X(R)
                                ;
011076  010046                  ADRMD6: MOV     R0,-(SP)        ;Save the register number
011100  106521                          MFPD    (R1)+           ;Get the vector base address
011102  062737  000002  000000G         ADD     #2,.D.ILEN      ;Say instruction uses 2 more bytes
011110  012600                          MOV     (SP)+,R0        ;Get value to R0
011112  004737  011570'                 CALL    ODCADR          ;Display the address
011116  012600                          MOV     (SP)+,R0        ;Recover the register number
011120  004737  010730'                 CALL    ADRMD1          ;Print "(r)"
```

```
     58 011124  000207                          RETURN
     59                                ;
     60                                ;  Mode 7 -- @X(R)
     61                                ;
     62 011126  010046                 ADRMD7:  MOV      R0,-(SP)        ;Save register number
     63 011130                                  .TTYOUT  #'@             ;Print "@"
     64 011140  012600                          MOV      (SP)+,R0        ;Recover register number
     65 011142  004737  011076'                 CALL     ADRMD6          ;Print "x(r)"
     66 011146  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Display routines for the various modes with R7.
                                ;
                                ;  Mode 2 -- #n
                                ;
011150                          AD7MD2: .TTYOUT #'#             ;Print "#"
011160  106521                          MFPD    (R1)+           ;Get the value onto the stack
011162  062737  000002  000000G         ADD     #2,D.ILEN       ;Say instruction uses 2 more bytes
011170  012600                          MOV     (SP)+,R0        ;Get the value
011172  004737  011672'                 CALL    ODCNEG          ;See if value should be printed as neg number
011176  103002                          BCC     9$              ;Br if it was a negative number
011200  004737  005772'                 CALL    PRTOCT          ;Print as an octal value
011204  000207                  9$:     RETURN
                                ;
                                ;  Mode 3 -- @#address
                                ;
011206                          AD7MD3: .TTYOUT #'@             ;Print "@#"
011216                                  .TTYOUT #'#
011226  106521                          MFPD    (R1)+           ;Get the address
011230  062737  000002  000000G         ADD     #2,D.ILEN       ;Advance the instruction length
011236  012600                          MOV     (SP)+,R0        ;Get the address
011240  004737  011570'                 CALL    ODCADR          ;Display the address
011244  000207                          RETURN
                                ;
                                ;  Mode 6 -- address
                                ;
011246  106521                  AD7MD6: MFPD    (R1)+           ;Get the address
011250  062737  000002  000000G         ADD     #2,D.ILEN       ;Advance the instruction length
011256  012600                          MOV     (SP)+,R0        ;Get the address
011260  060100                          ADD     R1,R0           ;Relocate the address
011262  004737  011570'                 CALL    ODCADR          ;Display the address
011266  000207                          RETURN
                                ;
                                ;  Mode 7 -- @address
                                ;
011270                          AD7MD7: .TTYOUT #'@             ;Print "@"
011300  000762                          BR      AD7MD6          ;Go display the address
```

```
      1                                 ;-----------------------------------------------------------------------
      2                                 ;  Jump table used to dispatch to the correct display routine for
      3                                 ;  addressing modes using registers R0 through R6.
      4                                 ;
      5 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
```

```
                                        .SBTTL  ODCREG -- Display register name
                                ;-----------------------------------------------------------------
                                ;  ODCREG is called to display the name of a register.
                                ;
                                ;  Inputs:
                                ;   R0 = Register number
                                ;
011342  010046                  ODCREG: MOV     R0,-(SP)
                                ;
                                ;  Determine which register this is
                                ;
011344  042700  177770                  BIC     #^C7,R0         ;Clear all but the register number
                                ;
                                ;  See if this is R7 (PC)
                                ;
011350  020027  000007                  CMP     R0,#7           ;Is register PC?
011354  001011                          BNE     1$
011356                                  .TTYOUT #'P             ;Display "PC"
011366                                  .TTYOUT #'C
011376  000427                          BR      9$
                                ;
                                ;  See if this is R6 (SP)
                                ;
011400  020027  000006          1$:     CMP     R0,#6           ;Is register SP?
011404  001011                          BNE     2$              ;Br if not
011406                                  .TTYOUT #'S             ;Print "SP"
011416                                  .TTYOUT #'P
011426  000413                          BR      9$
                                ;
                                ;  This is a register in the range R0 to R5.
                                ;
011430                          2$:     .TTYOUT #'R             ;Print "Rn"
011440  011600                          MOV     (SP),R0         ;Get register number
011442  042700  177770                  BIC     #^C7,R0
011446  062700  000060                  ADD     #'0,R0          ;Convert to ascii digit
011452                                  .TTYOUT                 ;Print it
                                ;
                                ;  Finished
                                ;
011456  062737  000002  000000G 9$:     ADD     #2,@#.PCOL      ;Advance print column counter
011464  012600                          MOV     (SP)+,R0
011466  000207                          RETURN
```

```
                                        .SBTTL  ODCACC -- Print a floating point accumulator name
                                ;-----------------------------------------------------------------------
                                ;  Print the name of a floating point accumulator.
                                ;
                                ;  Inputs:
                                ;   R0 = Accumulator number
                                ;
011470  010046                  ODCACC: MOV     R0,-(SP)
                                ;
                                ;  Print ACn
                                ;
011472                                  .TTYOUT #'A
011502                                  .TTYOUT #'C
011512  011600                          MOV     (SP),R0
011514  042700  177774                  BIC     #^C3,R0         ;Clear all but ACC #
011520  062700  000060                  ADD     #'0,R0          ;Convert to ascii char
011524                                  .TTYOUT                 ;Print the #
                                ;
                                ;  Finished
                                ;
011530  062737  000003  000000G         ADD     #3,D.PCOL       ;Advance print column number
011536  012600                          MOV     (SP)+,R0
011540  000207                          RETURN
```

```
                                        .SBTTL  ODCFPU -- Print general floating point operand
                                ;-----------------------------------------------------------------
                                ;  ODCFPU is called to display a general floating point operand.
                                ;
                                ;  Inputs:
                                ;   R0 = Floating point operand mode and register #.
                                ;
011542  010046                  ODCFPU: MOV     R0,-(SP)
                                ;
                                ;  Determine if it is a floating point accumulator
                                ;
011544  032700  000070                  BIT     #70,R0          ;Is mode = 0 (accumulator)?
011550  001003                          BNE     1$              ;Br if not
011552  004737  011470'                 CALL    ODCACC          ;Display an accumulator name
011556  000402                          BR      9$
                                ;
                                ;  It is not an accumulator
                                ;
011560  004737  010650'          1$:     CALL    ODCGEN          ;Display general operand
                                ;
                                ;  Finished
                                ;
011564  012600                  9$:     MOV     (SP)+,R0
011566  000207                          RETURN
```

```
                                        .SBTTL  ODCADR -- Display decoded instruction address
                                ;-----------------------------------------------------------------------
                                ;  ODCADR is called to display an address value which is an instruction
                                ;  operand.  If the address is relative to a relocation base, the
                                ;  address is displayed in the form "[r,offset]".
                                ;  If the address is not relative to a relocation base, it is displayed
                                ;  in the form "address".
                                ;
                                ;  Inputs:
                                ;   R0 = Address to be displayed.
                                ;
011570  010046                  ODCADR: MOV     R0,-(SP)
011572  010346                          MOV     R3,-(SP)
                                ;
                                ;  Determine if address should be shown as a negative number
                                ;
011574  004737  011672'                 CALL    ODCNEG          ;Should this be shown as a negative number?
011600  103031                          BCC     9$              ;Br if yes
                                ;
                                ;  Determine if address is relative to a relocation base
                                ;
011602  032737  0000000 0000000         BIT     #DP$DAA,D.PFMT  ;Should address be absolute?
011610  001023                          BNE     1$              ;Br if yes
011612  004737  007262'                 CALL    RELADR          ;See if address is within relocation region
011616  103420                          BCS     1$              ;Br if address is not within relocation region
                                ;
                                ;  Address is within a relocation region.
                                ;  Display in the form "[r,address]".
                                ;
011620  010003                          MOV     R0,R3           ;Save the address
011622                                  .TTYOUT #'[             ;Print "[r,address]"
011632  010300                          MOV     R3,R0           ;Get back address
011634  004737  007172'                 CALL    SHOADR          ;Display the address
011640                                  .TTYOUT #']             ;Put in closing bracket
011650  062737  000002  0000000         ADD     #2,D.PCOL       ;Advance print column counter
011656  000402                          BR      9$
                                ;
                                ;  Address is not relative to a relocation base
                                ;
011660  004737  005772'         1$:     CALL    PRTOCT          ;Display the address
                                ;
                                ;  Finished
                                ;
011664  012603                  9$:     MOV     (SP)+,R3
011666  012600                          MOV     (SP)+,R0
011670  000207                          RETURN
```

```
                                        .SBTTL  ODCNEG -- Determine if values should be shown as negative
                                ;-----------------------------------------------------------------------
                                ;  ODCNEG is called to determine if an address or value should be displayed
                                ;  as a negative number.
                                ;  Currently, the only values that are shown as negative numbers
                                ;  are -1 and -2.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to be tested.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Value was displayed as a negative number.
                                ;   C-flag set     ==> Value is not displayed as a negative number.
                                ;
011672                          ODCNEG:
                                ;
                                ;  See if this is a value to be displayed as a negative number
                                ;
011672  020027  177776                  CMP     R0,#177776      ;Is this a negative value?
011676  103414                  BLO     9$              ;Br if not
                                ;
                                ;  This is a negative value
                                ;
011700  010046                  MOV     R0,-(SP)
011702                          .TTYOUT #'-             ;Print minus sign
011712  011600                  MOV     (SP),R0         ;Get the value
011714  005400                  NEG     R0              ;Negate it
011716  004737  005772'          CALL    PRTOCT          ;Print it
011722  012600                  MOV     (SP)+,R0        ;Recover original value
011724  000241                  CLC                     ;This is a negative number
011726  000401                  BR      10$
                                ;
                                ;  Finished
                                ;
011730  000261                  9$:     SEC             ;This is not a negative value
011732  000207                  10$:    RETURN
```

```
                                        .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.
                                ;   R0     = Desired print column.
                                ;
                                ;  Outputs:
                                ;   D.PCOL = New print column after tabbing.
                                ;
011734  010046                  ODCTAB: MOV     R0,-(SP)
                                ;
                                ;  Print spaces until we reach the desired column
                                ;
011736                          1$:     .TTYOUT #SPACE          ;Print a space
011746  005237  0000000                 INC     D.PCOL          ;Advance column number
011752  023716  0000000                 CMP     D.PCOL,(SP)     ;Reached desired column?
011756  103767                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
011760  012600                          MOV     (SP)+,R0
011762  000207                          RETURN
```

```
                                .SBTTL  INSBAS -- Instruction decoding tables
                        ;-----------------------------------------------------------------------
                        ;  Table used for decoding a PDP-11 instruction into symbolic form.
                        ;  Each instruction is defined by an invocation of the INSTR macro
                        ;  which takes 4 arguments. The four arguments are:
                        ;    Argument 1 = 16 bit mask used to mask out the operand portions
                        ;                 of the instruction word.
                        ;    Argument 2 = Value of instruction after mask (arg 1) has been applied.
                        ;    Argument 3 = Operand type code for the instruction.
                        ;    Argument 4 = Symbolic name for the instruction.
                        ;
                        ;  Define symbolic names for offsets into an instruction description block:
                        ;
000000                  IB$MSK  =       0               ;Mask value
000002                  IB$VAL  =       2               ;Instruction value
000004                  IB$TYP  =       4               ;Operand type
000005                  IB$LEN  =       5               ;Length of instruction mnemonic text string
000006                  IB$NAM  =       6               ;Start of instruction mnemonic text string
                        ;
                                .MACRO  INSTR   MASK,VALUE,TYPE,NAME
                                .WORD   MASK
                                .WORD   VALUE
                                .BYTE   TYPE
                                .NCHR   NAMLEN,NAME
                                .BYTE   NAMLEN
                                .ASCII  /'NAME/
                                .EVEN
                                .ENDM   INSTR
                        ;
                        ;  Type codes for the instruction operand types:
                        ;    TYP1   = No operand.
                        ;    TYP2   = Single general operand.
                        ;    TYP4   = 4-bit absolute numeric operand.
                        ;    TYP5   = Condition code modification instruction.
                        ;    TYP6   = Branch offset operand.
                        ;    TYP7   = Register source, general operand destination.
                        ;    TYP9   = General source and destination operands.
                        ;    TYP10  = Register and loop offset (SOB).
                        ;    TYP11  = EMT and TRAP instructions.
                        ;    TYP12  = Single floating point general operand
                        ;    TYP13  = Floating point accumulator and floating source operand
                        ;    TYP14  = Floating point accumulator and general source operand
                        ;    TYP15  = Floating point accumulator and floating destination operand
                        ;    TYP16  = Floating point accumulator and general destination operand
                        ;
                        ;  Define the instructions
                        ;
011764                  INSBAS:
                        ;         Mask    Code    Type    Name
                        ;         ------  ------  ----    -----
011764                  INSTR   000000, 000000, TYP1,   HALT
011776                  INSTR   000000, 000001, TYP1,   WAIT
012010                  INSTR   000000, 000002, TYP1,   RTI
012022                  INSTR   000000, 000003, TYP1,   BPT
012034                  INSTR   000000, 000004, TYP1,   IOT
012046                  INSTR   000000, 000005, TYP1,   RESET
012062                  INSTR   000000, 000006, TYP1,   RTT
```

```
 58 012074
 59 012106
 60 012122
 61 012134
 62 012146
 63 012160
 64 012170
 65 012200
 66 012212
 67 012222
 68 012234
 69 012246
 70 012260
 71 012272
 72 012304
 73 012316
 74 012330
 75 012342
 76 012354
 77 012366
 78 012400
 79 012412
 80 012424
 81 012436
 82 012450
 83 012462
 84 012474
 85 012506
 86 012520
 87 012532
 88 012544
 89 012556
 90 012570
 91 012602
 92 012614
 93 012626
 94 012640
 95 012652
 96 012664
 97 012676
 98 012710
 99 012722
100 012734
101 012746
102 012760
103 012772
104 013004
105 013016
106 013030
107 013042
108 013054
109 013066
110 013100
111 013112
112 013124
113 013136
114 013150
```

```
INSTR   000077, 000100, TYP2,   JMP
INSTR   000000, 000207, TYP1,   RETURN
INSTR   000007, 000200, TYP3,   RTS
INSTR   000007, 000230, TYP4,   SPL
INSTR   000000, 000240, TYP1,   NOP
INSTR   000017, 000240, TYP5,   CL
INSTR   000017, 000260, TYP5,   SE
INSTR   000077, 000300, TYP2,   SWAB
INSTR   000377, 000400, TYP6,   BR
INSTR   000377, 001000, TYP6,   BNE
INSTR   000377, 001400, TYP6,   BEQ
INSTR   000377, 002000, TYP6,   BGE
INSTR   000377, 002400, TYP6,   BLT
INSTR   000377, 003000, TYP6,   BGT
INSTR   000377, 003400, TYP6,   BLE
INSTR   000077, 004700, TYP2,   CALL
INSTR   000777, 004000, TYP7,   JSR
INSTR   000077, 005000, TYP2,   CLR
INSTR   000077, 005100, TYP2,   COM
INSTR   000077, 005200, TYP2,   INC
INSTR   000077, 005300, TYP2,   DEC
INSTR   000077, 005400, TYP2,   NEG
INSTR   000077, 005500, TYP2,   ADC
INSTR   000077, 005600, TYP2,   SBC
INSTR   000077, 005700, TYP2,   TST
INSTR   000077, 006000, TYP2,   ROR
INSTR   000077, 006100, TYP2,   ROL
INSTR   000077, 006200, TYP2,   ASR
INSTR   000077, 006300, TYP2,   ASL
INSTR   000077, 006400, TYP8,   MARK
INSTR   000077, 006500, TYP2,   MFPI
INSTR   000077, 006600, TYP2,   MTPI
INSTR   000077, 006700, TYP2,   SXT
INSTR   007777, 010000, TYP9,   MOV
INSTR   007777, 020000, TYP9,   CMP
INSTR   007777, 030000, TYP9,   BIT
INSTR   007777, 040000, TYP9,   BIC
INSTR   007777, 050000, TYP9,   BIS
INSTR   007777, 060000, TYP9,   ADD
INSTR   000777, 070000, TYP8,   MUL
INSTR   000777, 071000, TYP8,   DIV
INSTR   000777, 072000, TYP8,   ASH
INSTR   000777, 073000, TYP8,   ASHC
INSTR   000777, 074000, TYP7,   XOR
INSTR   000007, 075000, TYP1,   FADD
INSTR   000007, 075010, TYP1,   FSUB
INSTR   000007, 075020, TYP1,   FMUL
INSTR   000007, 075030, TYP1,   FDIV
INSTR   000777, 077000, TYP10,  SOB
INSTR   000377, 100000, TYP6,   BPL
INSTR   000377, 100400, TYP6,   BMI
INSTR   000377, 101000, TYP6,   BHI
INSTR   000377, 101400, TYP6,   BLOS
INSTR   000377, 102000, TYP6,   BVC
INSTR   000377, 102400, TYP6,   BVS
INSTR   000377, 103000, TYP6,   BCC
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
```

```
                                TYPNAM  =       .-TYPVEC
                                        .WORD   RTN
                                        .ENDM   TYPDEF
                                ;
                                ;  Define the types
                                ;
                                TYPVEC::
                                        TYPDEF  TYP1,ODC1       ;No operand
                                        TYPDEF  TYP2,ODC2       ;Single general operand.
                                        TYPDEF  TYP3,ODC3       ;Single register operand (RTS)
                                        TYPDEF  TYP4,ODC4       ;4-bit absolute numeric operand.
                                        TYPDEF  TYP5,ODC5       ;Condition code modification instruction.
                                        TYPDEF  TYP6,ODC6       ;Branch offset operand.
                                        TYPDEF  TYP7,ODC7       ;Register source, general operand destination.
                                        TYPDEF  TYP8,ODC8       ;Register destination, general source operand
                                        TYPDEF  TYP9,ODC9       ;General source and destination operands.
                                        TYPDEF  TYP10,ODC10     ;Register and loop offset (SOB).
                                        TYPDEF  TYP11,ODC11     ;EMT and TRAP instructions.
                                        TYPDEF  TYP12,ODC12     ;One floating point operand.
                                        TYPDEF  TYP13,ODC13     ;Floating accum and floating source operand.
                                        TYPDEF  TYP14,ODC14     ;Floating accum and general source operand.
                                        TYPDEF  TYP15,ODC15     ;Floating accumulator and floating destination
                                        TYPDEF  TYP16,ODC16     ;Floating accumulator and general destination.

                                        .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
```

Cross reference table (CREF V05.04)

```
$DBGBK    1-33      6-16
$DEBUG    1-34      6-17
...V1    10-32     20-27     22-5      32-22     32-27     33-17     33-42     34-26     35-22     39-49     39-52     39-59
         41-27     45-55     47-29     47-32     47-35     47-38     48-41     49-15     50-16     51-16     52-54     53-15
         54-16     55-16     57-12     57-15     57-21     57-27     57-35     57-43     57-44     57-63     58-6      58-17
         58-18     58-36     60-18     60-19     60-26     60-27     60-32     60-36     61-12     61-13     61-17     63-31
         63-34     64-25     65-16
ACRNUM   29-42     29-54     30-10#
ACRVAL    9-20      9-31     29-15#
AD7MD2   58-6#     59-19
AD7MD3   58-17#    59-20
AD7MD6   58-27#    58-37     59-23
AD7MD7   58-36#    59-24
AD7VEC   56-34     59-17#
ADRMD0   57-6#     59-5      59-17
ADRMD1   57-11#    57-20     57-37     57-46     57-57     59-6      59-18
ADRMD2   57-20#    57-29     59-7
ADRMD3   57-26#    59-8
ADRMD4   57-34#    59-9      59-21
ADRMD5   57-42#    59-10     59-22
ADRMD6   57-51#    57-65     59-11
ADRMD7   57-62#    59-12
ADRVEC   56-39     59-5#
BELL      1-71#
BKSPAC    1-72#     2-21
BRKCHK    7-13     26-22#
BRKENT    1-21      6-29      7-8#
BRKRST    7-67     24-7#
BRKSET    7-24      8-14     25-6#
CHKADR   15-20     44-29#
CMDAT     2-30     12-6#
CMDAT1   12-7#     12-35
CMDB      2-35     15-5#
CMDBK1   10-22#    12-26     20-35     22-30
CMDBKS    2-29     10-8      10-21#    11-17     21-22
CMDBSP    2-45     22-5#
CMDCR     2-42     20-5#
CMDDCD    2-33     11-5#     21-8      21-17
CMDEXP    2-32     13-5#
CMDG      2-36     16-5#
CMDLF     2-43     18-58     20-17#
CMDM      2-37     19-5#
CMDP      2-38     16-16#    17-31
CMDR      2-39     14-5#
CMDRSB    2-34     21-5#
CMDS      2-40     16-26#
CMDSL1   10-7#     11-20     12-20     20-30     22-25
CMDSLH    2-28     10-6#
CMDTAB    2-46     17-9#
CMDTBL    2-4#      2-23      9-39
CMDUP     2-44     22-6      22-11#
CMDUS     2-31     12-32#
CMDVEC    2-28#     9-49
CMDX      2-41     18-5#
COLOPN    1-80#    46-14     46-33     47-8      48-8      48-33     49-8      50-8      51-8      52-8      52-28     52-47
         53-8      54-8      55-8
```

Cross reference table (CREF V05.04)

```
COMMA     1-75#   48-41   49-15   50-16   51-16   52-54   53-15   54-16   55-16
CORUSR    1-32     6-15   43-15
CR        1-69#    1-91    1-93    2-18   18-15   18-25   18-40   20-27   43-29
CRCHAR    1-93#   20-32   21-19
CRLF      1-92#    5-31    5-39    9-5    12-9    16-8    16-19   17-65   21-14   22-22   22-27   27-7
         27-28    27-41   27-56   27-68   35-6
D$BKST    1-30    24-8    24-26   25-21
D$CKBK    1-33    17-60   24-10   25-7    26-53   28-19   28-24
D$DBRK    1-28     5-43    7-14   26-26   26-49   27-11
D$DMON    1-29     7-22    7-65    8-6     8-12    8-28    8-54   19-5    19-8    26-30
D$DVAL    1-30    19-5    19-17   26-37
D$FBRK    1-31     6-28    7-14    7-77    8-5    27-54
D$IBRK    1-28     5-43    7-14    7-68   26-26   26-85   27-39
D$INIT    1-34     6-22   23-16
D$IPND    1-30     7-21    7-64    7-71    8-8     8-12    8-54   26-59
D$RUN     1-33     7-39    8-10
D$SBRK    1-28     5-43    7-14   26-26   26-66   27-32
D$SSTP    1-28     7-22    7-65    8-6     8-12    8-54   16-26   16-31   17-64   26-53   26-64
D$TSTP    1-34     7-22    7-65    7-84    8-6     8-12    8-54   17-30   26-53   26-64
D. BKAD   1-25     3-33   15-13   15-21*  24-16   25-13   26-75   28-12
D. BKNM   1-28    26-67*  26-84*  27-42   27-47
D. BKSV   1-29    24-21   25-16*
D. BYTM   1-25    10-9*   10-22*  20-24   21-11   22-19   37-26   37-56   38-22   38-46
D. CBRK   1-35    17-59*  27-44*
D. DADR   1-29     8-30   19-10*  26-32   27-17   27-20
D. DOLD   1-29     8-35*  26-40   27-23*  27-27*
D. DTRG   1-29    19-18*  26-44
D. END    1-31    23-13
D. FLAG   1-28     5-43*   6-22    6-28*   7-14    7-21*   7-22    7-39*   7-64*   7-65    7-68    7-71*
          7-77     7-84*   8-5*    8-6     8-8*    8-10*   8-12    8-28    8-54   16-26*  16-31*  17-30*
         17-60*   17-64*  19-5*   19-8*   19-17*  23-16*  24-8    24-10   24-26*  25-7    25-21*  26-26*
         26-30    26-37   26-49*  26-53   26-59   26-64   26-66*  26-85*  27-11   27-32   27-39   27-54
         28-19*   28-24*
D. ILEN   1-34    10-10*  10-23*  18-9*   21-13   45-20*  57-53*  58-8*   58-20*  58-28*
D. LOC    1-25     5-54*   8-9*   10-7    11-12   12-7*   12-33   13-22   13-29   20-10*  20-18   20-26*
         20-31*   21-9    21-13*  21-18*  22-12   22-21*  22-26*  36-16*  37-32   37-39   37-42   37-61*
         37-67    37-69   38-27   38-34   38-38   38-51   38-56   38-59   39-23   39-29
D. LOCM   1-26    11-18   12-8*   36-15*  37-16   37-21   38-17   39-18
D. LVAL   1-33    10-12*  10-25*  12-6    12-32   14-36   14-51   18-6
D. MASK   1-33     3-31    8-32   23-15*  26-34   27-25
D. NMBE   1-31    30-38
D. NMBF   1-25    30-28   30-54
D. PCNT   1-32     7-75    7-79*  16-9*   16-18*
D. PCOL   1-34    10-33*  11-28   18-53*  20-5*   20-17*  21-5*   22-11*  31-19*  32-23*  32-28*  33-18*
         33-43*   34-27*  35-8*   35-23*  39-50*  39-53*  39-60*  41-28*  43-24*  43-33*  45-25*  45-51*
         47-42*   48-42*  49-16*  51-17*  52-55*  53-16*  54-17*  55-17*  60-40*  61-21*  63-35*  65-17*
         65-18
D. PFMT   1-35     3-32   40-16   63-22
D. PS     1-30     3-30    4-17*   5-26*   7-35*   8-53*   8-56*   8-57*   8-61
D. R0     1-27     3-22    5-27*   7-9*    7-28    8-49
D. R1     1-27     3-23    7-40*   8-45
D. R2     1-27     3-24    7-41*   8-44
D. R3     1-27     3-25    7-42*   8-43
D. R4     1-27     3-26    5-24*   5-44    7-43*   8-42
D. R5     1-27     3-27    5-23*   5-45    7-44*   8-41
D. R6     1-27     3-28    7-46*   8-39
```

Cross reference table (CREF V05.04)

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

Cross reference table (CREF V05.04)

```
PSW       1-32     5-50#    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
```

Cross reference table (CREF V05.04)

```
. . . 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
```