Table of contents

Table of contents

```
                                        .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$LAA,D.CBRK
                                        .GLOBL  D$ISPC
                                ;
                                ;  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      111                         .BYTE   'I              ; I
000302      115                         .BYTE   'M              ; M
000303      120                         .BYTE   'P              ; P
000304      122                         .BYTE   'R              ; R
000305      123                         .BYTE   'S              ; S
000306      130                         .BYTE   'X              ; X
000307      015                         .BYTE   CR              ; Carriage-return
000310      012                         .BYTE   LF              ; Line-feed
000311      136                         .BYTE   '^              ; Up arrow
000312      010                         .BYTE   BKSPAC          ; Backspace
000313      011                         .BYTE   TAB             ; Tab
            000024              NUMCMD  =       .-CMDTBL
                                        .EVEN
                                ;
                                ;  Parallel table of command processing routines
                                ;
000314  001652'                 CMDVEC: .WORD   CMDSLH          ; /
000316  001730'                         .WORD   CMDBKS          ; \
000320  002136'                         .WORD   CMDAT           ; @
000322  002232'                         .WORD   CMDUS           ; _
000324  002252'                         .WORD   CMDEXP          ; !
000326  002036'                         .WORD   CMDDCD          ; [
000330  003730'                         .WORD   CMDRSB          ; ]
000332  002556'                         .WORD   CMDB            ; B
000334  002660'                         .WORD   CMDG            ; G
000336  003000'                         .WORD   CMDI            ; I
000340  003462'                         .WORD   CMDM            ; M
000342  002714'                         .WORD   CMDP            ; P
000344  002354'                         .WORD   CMDR            ; R
000346  002744'                         .WORD   CMDS            ; S
000350  003224'                         .WORD   CMDX            ; X
000352  003542'                         .WORD   CMDCR           ; Carriage-return
000354  003574'                         .WORD   CMDLF           ; Line-feed
000356  004072'                         .WORD   CMDUP           ; Up arrow
000360  004060'                         .WORD   CMDBSP          ; Backspace
000362  003034'                         .WORD   CMDTAB          ; Tab
```

```
                                ;-----------------------------------------------------------------------
                                ;  Table of internal register names
                                ;
 4 000364      060      INTCHR: .BYTE   '0              ; $0
 5 000365      061              .BYTE   '1              ; $1
 6 000366      062              .BYTE   '2              ; $2
 7 000367      063              .BYTE   '3              ; $3
 8 000370      064              .BYTE   '4              ; $4
 9 000371      065              .BYTE   '5              ; $5
10 000372      066              .BYTE   '6              ; $6 (SP)
11 000373      067              .BYTE   '7              ; $7 (PC)
12 000374      123              .BYTE   'S              ; $S (PS)
13 000375      115              .BYTE   'M              ; $M (Mask word)
14 000376      106              .BYTE   'F              ; $F (Format register)
15 000377      102              .BYTE   'B              ; $B (Breakpoint locations)
16 000400      122              .BYTE   'R              ; $R (Relocation base addresses)
17           000015     NUMINT  =       .-INTCHR
18                              .EVEN
19                              ;
20                              ;  Parallel table of addresses of internal register cells
21                              ;
22 000402  000000G      INTADR: .WORD   D.R0            ; $0
23 000404  000000G              .WORD   D.R1            ; $1
24 000406  000000G              .WORD   D.R2            ; $2
25 000410  000000G              .WORD   D.R3            ; $3
26 000412  000000G              .WORD   D.R4            ; $4
27 000414  000000G              .WORD   D.R5            ; $5
28 000416  000000G              .WORD   D.R6            ; $6 (SP)
29 000420  000000G              .WORD   D.R7            ; $7 (PC)
30 000422  000000G              .WORD   D.PS            ; $S (PS)
31 000424  000000G              .WORD   D.MASK          ; $M (Mask word)
32 000426  000000G              .WORD   D.PFMT          ; $F (Format register)
33 000430  000000G              .WORD   D.BKAD          ; $B (Breakpoint locations)
34 000432  000000G              .WORD   D.RLBS          ; $R (Relocation base addresses)
35 000434               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.
                                ;
000434                          DBGENT:
                                ;
                                ;  Initialize the debugger
                                ;
000434  004737  004226'                 CALL    DBGINI          ;Initialize the debugger
                                ;
                                ;  Set starting address and initial PSW
                                ;
000440  012637  000000G                 MOV     (SP)+,D.R7      ;Set starting address
000444  012637  000000G                 MOV     (SP)+,D.PS      ;Set starting PSW
                                ;
                                ;  Enter debugger
                                ;
000450  000137  001054'                 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
                                ;
000454                          DBGTRP:
                                ;
                                ;  See if the trap occurred in the debugger or in the user's program
                                ;
000454  005704                         TST     R4              ;Trap in debugger or user's program?
000456  001077                         BNE     10$             ;Br if trap occurred in the debugger
                                ;
                                ;  Trap occurred in the user's program.
                                ;  Save entry information.
                                ;
000460  012637  000000G                MOV     (SP)+,D.R5
000464  012637  000000G                MOV     (SP)+,D.R4
000470  012637  000000G                MOV     (SP)+,D.R7      ;Trap PC
000474  012637  000000G                MOV     (SP)+,D.PS      ;Trap PSW
000500  010037  000000G                MOV     R0,D.R0
                                ;
                                ;  Print trap entry message
                                ;
000504                                  TPRINT  #CRLF           ;Go to a new line
000514  020527  000001                  CMP     R5,#1           ;Trap to 4 or 10?
000520  001017                         BNE     1$              ;Br if trap to 10
000522                                  PRINT   <Trap to 4 at location >
000556  000416                         BR      2$
000560                          1$:     PRINT   <Trap to 10 at location >
000614  013700  000000G         2$:     MOV     D.R7,R0         ;Get address where trap occurred
000620  004737  007276'                 CALL    LSTADR          ;Display the address
000624                          4$:     TPRINT  #CRLF           ;Go to new line
                                ;
                                ;  Enter the debugger
                                ;
000634  042737  000000C 000000G        BIC     #<D$DBRK!D$IBRK!D$SBRK>,D.FLAG ;Say no breakpoint occurred
000642  013704  000000G                MOV     D.R4,R4         ;Restore R4 and R5
000646  013705  000000G                MOV     D.R5,R5
000652  000137  001054'                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.
                                ;
000656  013706  000000G         10$:    MOV     D.SPSV,SP       ;Restore the stack pointer
000662  052737  000000G 000000G        BIS     #UPMODE,@#PSW   ;Make sure previous mode = user
000670  005037  000000G                CLR     D.LOC           ;No currently open cell
000674  000137  001510'                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.
                ;
000700          DBGBRK:
                ;
                ;  Reset job control flag that was used to force the breakpoint
                ;
000700  010146          MOV     R1,-(SP)
000702  113701  000000G MOVB    CORUSR,R1       ;Get job index number
000706  042761  000000G 000000G BIC     #$DBGBK,LSW9(R1);Clear flag saying to force debugger break
000714  052761  000000G 000000G BIS     #$DEBUG,LSW9(R1);Say program is now running with debugger
000722  012601          MOV     (SP)+,R1
                ;
                ;  If we have not initialized the debugger, do the debugger startup
                ;
000724  032737  000000G 000000G BIT     #D$INIT,D.FLAG  ;Has initialization been done?
000732  001002          BNE     1$              ;Br if yes
000734  004737  004226' CALL    DBGINI          ;Initialize the debugger
                ;
                ;  Enter debugger as if we hit a breakpoint
                ;
000740  052737  000000G 000000G 1$:     BIS     #D$FBRK,D.FLAG  ;Set flag saying we had a forced break
000746  000137  000752' 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.
                                ;
000752  011637  000000G       BRKENT: MOV     (SP),D.R7       ;Save breakpoint PC
000756  010037  000000G               MOV     R0,D.R0         ;Save user's R0
                                ;
                                ;  Determine if we should enter the debugger
                                ;
000762  004737  004456'               CALL    BRKCHK          ;See if breakpoint causes program stop
000766  032737  000000C 000000G       BIT     #<D$DBRK!D$IBRK!D$SBRK!D$FBRK>,D.FLAG ;Breakpoint?
000774  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.
                                ;
000776  042737  000000G 000000G       BIC     #D$IPND,D.FLAG  ;Say pending instruction has been executed
001004  032737  000000C 000000G       BIT     #<D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Are we single stepping
001012  001006                       BNE     3$              ;Br if yes
001014  004737  004376'               CALL    BRKSET          ;Set breakpoints
001020  042766  000020  000002       BIC     #TRCTRP,2(SP)   ;Reset trace trap flag in PSW
001026  000403                       BR      2$
001030  052766  000020  000002  3$:  BIS     #TRCTRP,2(SP)   ;Set trap flag in PSW on stack
001036  013700  000000G       2$:     MOV     D.R0,R0         ;Recover R0
001042  000006                       RTT                     ;Return from breakpoint
                                ;
                                ;  This breakpoint will enter the debugger.
                                ;  Save information and enter the debugger
                                ;
001044  012637  000000G       5$:     MOV     (SP)+,D.R7      ;Save trap PC
001050  012637  000000G               MOV     (SP)+,D.PS      ;Save trap PS
                                ;
                                ;  Save user's registers
                                ;
001054  052737  000000G 000000G DBGRUN: BIS     #D$RUN,D.FLAG   ;Set flag saying debugger is running
001062  010137  000000G               MOV     R1,D.R1
001066  010237  000000G               MOV     R2,D.R2
001072  010337  000000G               MOV     R3,D.R3
001076  010437  000000G               MOV     R4,D.R4
001102  010537  000000G               MOV     R5,D.R5
001106  106506                       MFPD    SP
001110  012637  000000G               MOV     (SP)+,D.R6
                                ;
                                ;  Save initial stack pointer in case we need to restore it if an
                                ;  error occurs.
                                ;
001114  010637  000000G               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.
                                ;
001120  106537  000036               MFPD    @#36            ;Get user's TRAP PSW
```

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

```
                                        .SBTTL  EXIT   -- Exit from debugger to user
                                ;-------------------------------------------------------------------------
                                ;  Exit from debugger to user's program.
                                ;
001250  042737  000000G 000000G EXIT1:  BIC     #D$FBRK,D.FLAG  ;Turn off forced-breakpoint flag
001256  032737  000000C 000000G         BIT     #<D$SSTP!D$TSTP!D$DMON>,D.FLAG ;Are we single stepping?
001264  001403                         BEQ     3$              ;Br if not
001266  042737  000000G 000000G         BIC     #D$IPND,D.FLAG  ;Turn of instruction-pending if single step
001274  005037  000000G         3$:     CLR     D.LOC           ;Say no location is open
001300  042737  000000G 000000G         BIC     #D$RUN,D.FLAG   ;Say debugger is no longer running
001306  004737  007624'                 CALL    DBGOFF          ;Turn off terminal debug mode
001312  032737  000000C 000000G         BIT     #<D$IPND!D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Should we single step?
001320  001003                         BNE     4$              ;Br if yes
001322  004737  004376'                 CALL    BRKSET          ;Set breakpoints
001326  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.
                                ;
001330  106537  000036          4$:     MFPD    @#36            ;Get user's TRAP PSW
001334  052716  000020                  BIS     #TRCTRP,(SP)    ;Set the trace-trap flag in the PSW
001340  106637  000036                  MTPD    @#36            ;Store new TRAP PSW
                                ;
                                ;  If we are doing data monitoring, save current value of monitored cell
                                ;
001344  032737  000000G 000000G 1$:     BIT     #D$DMON,D.FLAG  ;Are we doing data cell monitoring?
001352  001411                         BEQ     2$              ;Br if not
001354  013705  000000G                 MOV     D.DADR,R5       ;Get address of cell being monitored
001360  106515                         MFPD    (R5)            ;Get current value from cell
001362  013704  000000G                 MOV     D.MASK,R4       ;Get current data mask
001366  005104                         COM     R4              ;Complement
001370  040405                         BIC     R4,R5           ;Clear all but bits of interest
001372  012637  000000G                 MOV     (SP)+,D.DOLD    ;Save old value
                                ;
                                ;  Restore user's registers
                                ;
001376  013746  000000G         2$:     MOV     D.R6,-(SP)
001402  106606                         MTPD    SP
001404  013705  000000G                 MOV     D.R5,R5
001410  013704  000000G                 MOV     D.R4,R4
001414  013703  000000G                 MOV     D.R3,R3
001420  013702  000000G                 MOV     D.R2,R2
001424  013701  000000G                 MOV     D.R1,R1
                                ;
                                ;  Exit point used if we did not enter debugger
                                ;
001430  013700  000000G         EXIT2:  MOV     D.R0,R0
                                ;
                                ;  See if we need to set Trap flag in PS
                                ;
001434  042737  000020  000000G         BIC     #TRCTRP,D.PS    ;Reset trace trap flag in PS
001442  032737  000000C 000000G         BIT     #<D$IPND!D$DMON!D$SSTP!D$TSTP>,D.FLAG ;Do we want single step?
001450  001403                         BEQ     1$              ;Br if not
001452  052737  000020  000000G         BIS     #TRCTRP,D.PS    ;Set Trap flag in user's PS
001460  052737  000000C 000000G 1$:     BIS     #UMODE!UPMODE,D.PS ;Make sure user-mode is set in PS
```

```
                                        ;
                                        ;  Use RTT instruction to reenter user's program
                                        ;
001466  013746  000000G                 MOV     D.PS,-(SP)      ;Set PS for user
001472  013746  000000G                 MOV     D.R7,-(SP)      ;Set PC for user
001476  000006                          RTT                     ;Enter user's program
```

```
                                        .SBTTL  NEWCMD -- Get new command
                                ;-------------------------------------------------------------------------
                                ;  Print carriage-return, line-feed and get new command
                                ;
001500                          NEWLIN: TPRINT  #CRLF
                                ;
                                ;  Get new command
                                ;
001510                          NEWCMD: PRINT   <DBG:>
                                ;
                                ;  Assume neither value1 nor value2 is specified
                                ;
001522  105037  000000G        GETCMD: CLRB    D.V1FL          ;Value1 not specified
001526  005037  000000G                CLR     D.VAL1
001532  105037  000000G                CLRB    D.V2FL          ;Value2 not specified
001536  005037  000000G                CLR     D.VAL2
                                ;
                                ;  See if Value1 is specified
                                ;
001542  004737  005310'                 CALL    ACRVAL          ;Try to accrue value1
001546  010037  000000G                MOV     R0,D.VAL1       ;Save value for value1
001552  110137  000000G                MOVB    R1,D.V1FL       ;Remember if value1 specified
                                ;
                                ;  See if value2 is specified
                                ;
001556  004737  005756'                 CALL    GETCHR          ;Get delimiter
001562  120027  000073                 CMPB    R0,#';          ;is there a value2?
001566  001403                        BEQ     1$              ;Br if possibly yes
001570  004737  006022'                 CALL    PSHCHR          ;Save command character
001574  000406                        BR      2$
001576  004737  005310'         1$:    CALL    ACRVAL          ;Get value2
001602  010037  000000G                MOV     R0,D.VAL2       ;Save value2
001606  110137  000000G                MOVB    R1,D.V2FL       ;Save info about value2
                                ;
                                ;  Look up command character
                                ;
001612  004737  005756'         2$:    CALL    GETCHR          ;Get command character
001616  012701  000024                MOV     #NUMCMD,R1      ;Get # valid command characters
001622  120061  000270'         3$:    CMPB    R0,CMDTBL(R1)   ;Search for character in table
001626  001406                        BEQ     4$              ;Br if found command character
001630  005301                        DEC     R1              ;More command chars to check?
001632  002373                        BGE     3$              ;Loop if yes
001634                                  ERR     #EM$IVC         ;Invalid command character
                                ;
                                ;  Found command character.
                                ;  Jump off to processing routine.
                                ;
001644  006301                  4$:    ASL     R1              ;Convert command index into word table index
001646  000171  000314'                 JMP     @CMDVEC(R1)     ;Jump to processing routine
```

```
                                        .SBTTL  -- Command processing routines --
                                        .SBTTL  "/"  -- Examine word
                                ;-----------------------------------------------------------------
                                ;  "address/" -- Examine contents of word
                                ;
001652  004737  006426'         CMDSLH: CALL    SETLOC          ;Set value1 as current location
001656  032737  000001  000000G CMDSL1: BIT     #1,D.LOC        ;Is location odd?
001664  001021                         BNE     CMDBKS          ;If yes then treat "/" like "\"
001666  105037  000000G                CLRB    D.BYTM          ;Say we are in word mode
001672  012737  000002  000000G        MOV     #2,D.ILEN       ;Say instruction length = 2 bytes
001700  004737  006714'                CALL    GETVAL          ;Get contents of location
001704  010037  000000G                MOV     R0,D.LVAL       ;Save last value displayed
001710  004737  006136'                CALL    PRTWRD          ;Print the value
001714                                 PRINT   <  >            ;Print two spaces
001724  000137  001522'                JMP     GETCMD          ;Go get next command

                                        .SBTTL  "\"  -- Examine byte
                                ;-----------------------------------------------------------------
                                ;  "address\" -- Examine contents of a byte
                                ;
001730  004737  006426'         CMDBKS: CALL    SETLOC          ;Set value1 as current location
001734  112737  000001  000000G CMDBK1: MOVB    #1,D.BYTM       ;Say we are operating in byte mode
001742  112737  000001  000000G        MOVB    #1,D.ILEN       ;Say instruction length = 1 byte
001750  004737  006714'                CALL    GETVAL          ;Get contents of location
001754  010037  000000G                MOV     R0,D.LVAL       ;Save last value displayed
001760  010002                         MOV     R0,R2           ;Save value
001762  004737  006212'                CALL    PRTBYT          ;Print value
001766                                 PRINT   <=>             ;Put in equal sign
001776  120227  000040                 CMPB    R2,#40          ;Is this a printing character?
002002  103002                         BHIS    1$              ;Br if yes
002004  112702  000056                MOVB    #'.,R2          ;Use period for non-printing chars
002010                          1$:    .TTYOUT R2              ;Print character
002016  005237  000000G                INC     D.PCOL          ;Advance print column
002022                                 PRINT   <  >            ;Print 2 spaces
002032  000137  001522'                JMP     GETCMD          ;Go get next command
```

```
                                        .SBTTL  "[" -- Decode value as an instruction
                                ;-------------------------------------------------------------------
                                ;  Open a cell and display its contents as an instruction.
                                ;
002036                          CMDDCD:
                                ;
                                ;  See if an address was specified
                                ;
002036  105737  0000006G               TSTB    D.V1FL          ;Was an address specified?
002042  001402                         BEQ     1$              ;Br if not
002044  004737  006426'               CALL    SETLOC          ;Set address of currently open cell
002050  013701  0000006G       1$:     MOV     D.LOC,R1        ;Is a cell currently open?
002054  001002                         BNE     2$              ;Br if yes
002056  000137  001500'               JMP     NEWLIN          ;Nothing to decode
002062  032701  000001         2$:     BIT     #1,R1           ;Is the address odd?
002066  001402                         BEQ     3$              ;Br if not
002070  000137  001730'               JMP     CMDBKS          ;Treat like "\" if odd
002074  105737  0000006G       3$:     TSTB    D.LOCM          ;Is cell internal or external?
002100  002002                         BGE     4$              ;Br if external
002102  000137  001656'               JMP     CMDSL1          ;Treat [ like / for internal cell
                                ;
                                ;  Decode the instruction
                                ;
002106  004737  007640'         4$:     CALL    DECODE          ;Decode the instruction
                                ;
                                ;  Tab over some
                                ;
002112  013700  0000006G               MOV     D.PCOL,R0       ;Get current print column
002116  062700  000010                 ADD     #8.,R0          ;Tab over some
002122  042700  000007                 BIC     #7,R0
002126  004737  012064'               CALL    ODCTAB          ;Tab over
                                ;
                                ;  Finished
                                ;
002132  000137  001522'               JMP     GETCMD
```

```
                                        .SBTTL  "@"  -- Open indirect
                                ;-------------------------------------------------------------------
                                ;  Open the cell whose address is contained in the cell pointed to
                                ;  by D.LOC.
                                ;
002136  013702  000000G         CMDAT:  MOV     D.LVAL,R2       ;Get last displayed value
002142  010237  000000G         CMDAT1: MOV     R2,D.LOC        ;Set as new location to display
002146  112737  000001  000000G        MOVB    #1,D.LOCM       ;Say new address is in user's program
002154                                  TPRINT  #CRLF           ;Go to a new line
                                ;
                                ;  See if we should do the display in byte or word mode
                                ;
002164  032702  000001                  BIT     #1,R2           ;Should we display in byte or word mode?
002170  001010                          BNE     1$              ;Br if need to go to byte mode
                                ;
                                ;  Display new location in word mode
                                ;
002172  004737  007100'                 CALL    SHOLOC          ;Display the current address
002176                                  PRINT   < / >           ;Display "/"
002206  000137  001656'                 JMP     CMDSL1          ;Simulate the "/" command
                                ;
                                ;  Display in byte mode
                                ;
002212  004737  007100'         1$:     CALL    SHOLOC          ;Display the current address
002216                                  PRINT   < \ >           ;Display "\"
002226  000137  001734'                 JMP     CMDBK1          ;Simulate the "\" command

                                        .SBTTL  "_"  -- Open cell indirect, relative to PC
                                ;-------------------------------------------------------------------
                                ;  "_" -- Open indirect relative to PC.
                                ;
002232  013702  000000G         CMDUS:  MOV     D.LVAL,R2       ;Get last displayed value
002236  063702  000000G                 ADD     D.LOC,R2        ;Add PC of instruction
002242  062702  000002                  ADD     #2,R2           ;Plus 2
002246  000137  002142'                 JMP     CMDAT1          ;Now treat like "@" command
```

```
                                        .SBTTL  "!"  -- Convert address to relocation offset
                                ;-------------------------------------------------------------------
                                ;  "n!" -- Convert current address to relocation offset
                                ;
002252                          CMDEXP:
                                ;
                                ;  If no relocation register number was provided, find closest relocation
                                ;  base.
                                ;
002252  013701  000000G               MOV     D.VAL1,R1       ;Get relocation register #
002256  105737  000000G               TSTB    D.V1FL          ;Was a relocation register specified?
002262  003422                        BLE     2$              ;Br if not
                                ;
                                ;  A relocation register number was provided.
                                ;  Display the relocation register number and the offset.
                                ;
002264                                PRINT   <=>             ;Print "="
002274  010100                        MOV     R1,R0           ;Get the relocation register number
002276  004737  006030'               CALL    PRTOCT          ;Print the relocation reg #
002302                                PRINT   <,>             ;Print ","
002310  006301                        ASL     R1              ;Convert relocation # to word table index
002312  013700  000000G               MOV     D.LOC,R0        ;Get original value
002316  166100  000000G               SUB     D.RLBS(R1),R0   ;Subtract relocation base
002322  004737  006030'               CALL    PRTOCT          ;Print the offset
002326  000404                        BR      8$
                                ;
                                ;  No relocation register number was specified
                                ;
002330  013700  000000G       2$:     MOV     D.LOC,R0        ;Get address
002334  004737  007322'               CALL    SHOADR          ;Display it with best relocation base
002340                        8$:     PRINT   <  >            ;Print 2 spaces
                                ;
                                ;  Finished
                                ;
002350  000137  001522'       9$:     JMP     GETCMD          ;Go get the next command
```

```
                                        .SBTTL  "R"  -- Set relocation base register
                                ;-----------------------------------------------------------------------
                                ;  "address;nR" -- Set relocation base register
                                ;
002354                          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.
                                ;
002354  105737  000000G                TSTB    D.V2FL          ;Was argument 2 specified?
002360  001417                        BEQ     2$              ;Br if not
                                ;
                                ;  Set base address for relocation area
                                ;
002362  013701  000000G                MOV     D.VAL2,R1       ;Get register number
002366  020127  000007                CMP     R1,#7           ;Should not exceed 7
002372  101404                        BLOS    1$              ;Br if ok
002374                                ERR     #EM$IVL
002404  006301                1$:     ASL     R1              ;Convert to word table index
002406  013761  000000G 000000G        MOV     D.VAL1,D.RLBS(R1);Set address for relocation base
002414  000137  001500'                JMP     NEWLIN          ;Go get next command
                                ;
                                ;  Calculate offset relative to relocation base
                                ;
002420                          2$:     PRINT   <=>             ;Print "="
002430  105737  000000G                TSTB    D.V1FL          ;Was a relocation reg # provided?
002434  001436                        BEQ     4$              ;Br if not
002436  013701  000000G                MOV     D.VAL1,R1       ;Get register number
002442  020127  000007                CMP     R1,#7           ;Should not exceed 7
002446  101404                        BLOS    3$              ;Br if ok
002450                                ERR     #EM$IVL
002460  006301                3$:     ASL     R1              ;Convert to word table index
002462  013700  000000G                MOV     D.LVAL,R0       ;Get currently open value
002466  166100  000000G                SUB     D.RLBS(R1),R0   ;Calculate offset
002472  010002                        MOV     R0,R2           ;Hold the offset value
002474  010100                        MOV     R1,R0           ;Get offset region #
002476  006200                        ASR     R0              ;Convert to ordinal number
002500  004737  006030'                CALL    PRTOCT          ;Print relocation #
002504                                PRINT   <,>             ;Print comma
002512  010200                        MOV     R2,R0           ;Get offset value
002514  004737  006030'                CALL    PRTOCT          ;Print it
002520                                PRINT   <  >            ;Print a space
002530  000410                        BR      9$
                                ;
                                ;  No relocation register number was provided
                                ;  Show address with best relocation
                                ;
002532  013700  000000G        4$:     MOV     D.LVAL,R0       ;Get value of interest
002536  004737  007322'                CALL    SHOADR          ;Display the address
002542                                PRINT   <  >            ;Print 2 spaces
                                ;
                                ;  Finished
                                ;
002552  000137  001522'        9$:     JMP     GETCMD          ;Go get next command
```

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

```
                                        .SBTTL  "G"  -- Start program execution
                                ;-------------------------------------------------------------------
                                ;  "address;G" -- Start program execution.
                                ;
002660  105737  000000G         CMDG:   TSTB    D.V1FL          ;Was a starting address specified?
002664  003403                          BLE     1$              ;Br if not
002666  013737  000000G 000000G         MOV     D.VAL1,D.R7     ;Set starting address
002674                          1$:     TPRINT  #CRLF           ;Go to new line
002704  005037  000000G                 CLR     D.PCNT          ;Clear any proceed count
002710  000137  001250'                 JMP     EXIT1           ;Start running program

                                        .SBTTL  "P"  -- Proceed from breakpoint
                                ;-------------------------------------------------------------------
                                ;  "n;P" -- Proceed from breakpoint
                                ;
002714  105737  000000G         CMDP:   TSTB    D.V1FL          ;Was a proceed repeat count specified?
002720  003403                          BLE     1$              ;Br if not
002722  013737  000000G 000000G         MOV     D.VAL1,D.PCNT   ;Set proceed repeat count
002730                          1$:     TPRINT  #CRLF           ;Go to new line
002740  000137  001250'                 JMP     EXIT1           ;Enter user's program

                                        .SBTTL  "S"  -- Set/Reset single-step mode
                                ;-------------------------------------------------------------------
                                ;  ";nS" -- Set or reset single-step mode
                                ;
002744  042737  000000G 000000G CMDS:   BIC     #D$SSTP,D.FLAG  ;Assume he wants to reset single-step mode
002752  105737  000000G                 TSTB    D.V2FL          ;Was a value specified for "n"?
002756  001406                          BEQ     1$              ;Br if not
002760  005737  000000G                 TST     D.VAL2          ;Is value 2 non-zero?
002764  001403                          BEQ     1$              ;If zero, don't single step
002766  052737  000000G 000000G         BIS     #D$SSTP,D.FLAG  ;Set single-step mode
002774  000137  001500'         1$:     JMP     NEWLIN          ;Go get next command

                                        .SBTTL  "I"  -- Set/Reset I-space value inspection
                                ;-------------------------------------------------------------------
                                ;  ";nI" -- Set or reset I-space value inspection (for / and \)
                                ;  I-space is always used for instruction decoding
                                ;
003000  042737  000000G 000000G CMDI:   BIC     #D$ISPC,D.FLAG  ;Assume / and \ refer to D-space
003006  105737  000000G                 TSTB    D.V2FL          ;Was a value specified for "n"?
003012  001406                          BEQ     1$              ;Br if not
003014  005737  000000G                 TST     D.VAL2          ;Was value zero?
003020  001403                          BEQ     1$              ;0 or no value mean use D-space
003022  052737  000000G 000000G         BIS     #D$ISPC,D.FLAG  ;Set I-space for / and \
003030  000137  001500'         1$:     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.
                                ;
003034                          CMDTAB:
                                ;
                                ;  See if a value was specified with TAB
                                ;
003034  105737  000000G                 TSTB    D.V1FL          ;Was value 1 specified?
003040  001407                          BEQ     2$              ;Br if not
003042  105037  000000G                 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.
                                ;
003046  013700  000000G         1$:     MOV     D.VAL1,R0       ;Get specified value
003052  001407                          BEQ     5$              ;0 ==> Skip over subprogram call
                                ;
                                ;  A starting address was specified.
                                ;
003054  010037  000000G                 MOV     R0,D.R7         ;Set starting address
                                ;
                                ;  Single step the program.
                                ;
003060  052737  000000G 000000G 2$:     BIS     #D$TSTP,D.FLAG  ;Say we are in temporary single step mode
003066  000137  002714'                 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.
                                ;
003072  013701  000000G         5$:     MOV     D.R7,R1         ;Get address of current instruction
003076  006521                          MFPI    (R1)+           ;Get instruction from user's program
003100  012602                          MOV     (SP)+,R2        ;Get the instruction from the stack
003102  010200                          MOV     R2,R0           ;Copy the instruction word
003104  042700  000777                  BIC     #777,R0         ;Clear register # and address field
003110  020027  004000                  CMP     R0,#0004000     ;Is this a JSR instruction?
003114  001361                          BNE     2$              ;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.
                                ;
003116  010200                          MOV     R2,R0           ;Get the instruction
003120  042700  177770                  BIC     #^C7,R0         ;Clear all but addressing register #
003124  020027  000007                  CMP     R0,#7           ;Is it using register 7 (PC)?
003130  001003                          BNE     3$              ;Br if not
003132  062701  000002                  ADD     #2,R1           ;2 bytes for subroutine address if PC used
003136  000410                          BR      4$
003140  010200                  3$:     MOV     R2,R0           ;Get back the instruction
003142  042700  177707                  BIC     #^C70,R0        ;Clear all but addressing mode
003146  072027  177775                  ASH     #-3,R0          ;Right justify the addressing mode
003152  116000  003214'                 MOVB    INSLEN(R0),R0   ;Get # bytes used by this addressing mode
```

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

```
                                        .SBTTL  "X"  -- Display RAD50 value
                                ;-------------------------------------------------------------------
                                ;  "X" -- Convert currently displayed value from RAD50 to ascii.
                                ;
003224                          CMDX:   PRINT   <=>             ;Print "="
003234  013700  000000G                 MOV     D.LVAL,R0       ;Get last displayed value
003240  004737  006270'                 CALL    PRTR50          ;Convert to ascii and print
003244                                  PRINT   <  >            ;Print spaces
003254  012737  000002  000000G         MOV     #2,D.ILEN       ;Say instruction length = 2 bytes
                                ;
                                ;  See if user wants to enter a new RAD50 value for this cell
                                ;
003262  004737  005756'                 CALL    GETCHR          ;Get 1st character
003266  004737  006022'                 CALL    PSHCHR          ;Push the character
003272  120027  000015                  CMPB    R0,#CR          ;No new value wanted?
003276  001454                          BEQ     7$              ;Br if no new value specified
003300  120027  000012                  CMPB    R0,#LF          ;Advance to next cell?
003304  001451                          BEQ     7$              ;Br if yes
                                ;
                                ;  Now accept a new RAD50 value
                                ;
003306  005001                          CLR     R1              ;Form new RAD50 value in R1
003310  012702  000003                  MOV     #3,R2           ;Set count of # chars to accrue
003314  004737  005756'         1$:     CALL    GETCHR          ;Get next character of value
003320  120027  000015                  CMPB    R0,#CR          ;Reached end?
003324  001434                          BEQ     2$              ;Br if yes
003326  120027  000012                  CMPB    R0,#LF          ;LF is also end
003332  001431                          BEQ     2$
003334  012703  000220'                 MOV     #R50CHR,R3      ;Point to RAD50 character table
003340  120023                  3$:     CMPB    R0,(R3)+        ;Search for char in table
003342  001407                          BEQ     4$              ;Br if found
003344  020327  000270'                 CMP     R3,#R50CHR+50   ;Checked all of table?
003350  103773                          BLO     3$              ;Loop if not
003352                                  ERR     #EM$IRC         ;Invalid RAD50 character
003362  162703  000221'         4$:     SUB     #R50CHR+1,R3    ;Get RAD50 character value
003366  070127  000050                  MUL     #50,R1          ;Multiply previous value by 50
003372  060301                          ADD     R3,R1           ;Add new character value
003374  077231                          SOB     R2,1$           ;Loop if more characters needed
003376  004737  005756'         6$:     CALL    GETCHR          ;Ignore other characters up to CR
003402  120027  000015                  CMPB    R0,#CR
003406  001403                          BEQ     2$
003410  120027  000012                  CMPB    R0,#LF
003414  001370                          BNE     6$
                                ;
                                ;  We have gotten the RAD50 value.  Store it in currently open cell
                                ;
003416  004737  006022'         2$:     CALL    PSHCHR          ;Save the terminating character
003422  010100                          MOV     R1,R0           ;Get value to R0 for STRVAL
003424  004737  006450'                 CALL    STRVAL          ;Store value into currently open cell
                                ;
                                ;  If command ended with Line feed, open next cell
                                ;
003430  005037  000000G         7$:     CLR     D.PCOL          ;Say print column = 0
003434  004737  005756'                 CALL    GETCHR          ;Get the terminating character
003440  120027  000012                  CMPB    R0,#LF          ;Open the next cell?
003444  001004                          BNE     9$              ;Br if not
003446  105037  000000G                 CLRB    D.V1FL          ;Tell LF not to store into current cell
```

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

```
                                        .SBTTL  "M"  -- Monitor data cell
                                ;-------------------------------------------------------------------
                                ;  "address;valueM" -- Monitor data cell
                                ;
003462  042737  000000C 000000G CMDM:   BIC     #<D$DMON!D$DVAL>,D.FLAG ;Reset data-cell monitoring flags
003470  005737  000000G                 TST     D.VAL1          ;Was an address specified?
003474  001420                          BEQ     9$              ;Br if not -- Disable monitoring
003476  052737  000000G 000000G         BIS     #D$DMON,D.FLAG  ;Enable data monitoring
003504  013700  000000G                 MOV     D.VAL1,R0       ;Get address of cell to monitor
003510  010037  000000G                 MOV     R0,D.DADR       ;Set it as control address
                                ;
                                ;  See if breakpoint is to be triggered whenever value in cell changes
                                ;  or only when a specified value occurs.
                                ;
003514  105737  000000G                 TSTB    D.V2FL          ;Was a value specified?
003520  003406                          BLE     9$              ;Br if not
003522  052737  000000G 000000G         BIS     #D$DVAL,D.FLAG  ;Watch for specified value
003530  013737  000000G 000000G         MOV     D.VAL2,D.DTRG   ;Save the specified value
                                ;
                                ;  Finished
                                ;
003536  000137  001500'          9$:     JMP     NEWLIN          ;Go get next command
```

```
                                        .SBTTL  "CR" -- Close current location
                                ;-------------------------------------------------------------------
                                ;  "carriage-return" -- Close current cell
                                ;
003542  005037  000000G         CMDCR:  CLR     D.PCOL          ;Say print column = 0
003546  105737  000000G                 TSTB    D.V1FL          ;Was a new value specified for this cell?
003552  003404                          BLE     9$              ;Br if not
003554  013700  000000G                 MOV     D.VAL1,R0       ;Yes, get the value
003560  004737  006450'                 CALL    STRVAL          ;Store value into current location
003564  005037  000000G         9$:     CLR     D.LOC           ;Say no location open now
003570  000137  001510'                 JMP     NEWCMD          ;Go get next command

                                        .SBTTL  "LF" -- Close location and advance to next
                                ;-------------------------------------------------------------------
                                ;  "line-feed" -- Close current cell and advance to next
                                ;
003574  005037  000000G         CMDLF:  CLR     D.PCOL          ;Say print column = 0
003600  005737  000000G                 TST     D.LOC           ;Is a location open now?
003604  001447                          BEQ     5$              ;Br if not
003606  105737  000000G                 TSTB    D.V1FL          ;Was a new value specified for this cell?
003612  003404                          BLE     1$              ;Br if not
003614  013700  000000G                 MOV     D.VAL1,R0       ;Get new value
003620  004737  006450'                 CALL    STRVAL          ;Store into currently open cell
003624  105737  000000G         1$:     TSTB    D.BYTM          ;Are we in byte or word mode?
003630  001017                          BNE     2$              ;Br if in byte mode
003632  062737  000002  000000G        ADD     #2,D.LOC        ;Advance location counter
003640                                  .TTYOUT #CR             ;Go to new line
003650  004737  007100'                 CALL    SHOLOC          ;Display address of new location
003654                                  PRINT   </>             ;Display "/"
003664  000137  001656'                 JMP     CMDSL1          ;Simulate "/" command
003670  005237  000000G         2$:     INC     D.LOC           ;Advance to next byte
003674                                  TPRINT  #CRCHAR         ;go to new line
003704  004737  007100'                 CALL    SHOLOC          ;Display address of new location
003710                                  PRINT   <\>             ;Display "\"
003720  000137  001734'                 JMP     CMDBK1          ;Simulate "\" command
003724  000137  001500'         5$:     JMP     NEWLIN          ;No cell open -- Ignore LF
```

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

```
                                        .SBTTL  "^"  -- Close current location and go to previous
                                ;-------------------------------------------------------------------
                                ;  "Backspace" -- Equivalent to up-arrow
                                ;
004060                          CMDBSP: .TTYOUT #136            ;Print "^"
004070  000400                          BR      CMDUP           ;Now treat backspace like up-arrow

                                ;-------------------------------------------------------------------
                                ;  "^" -- Close current location and go to previous one.
                                ;
004072  005037  000000G         CMDUP:  CLR     D.PCOL          ;Say print column = 0
004076  005737  000000G                 TST     D.LOC           ;Is a cell open?
004102  001002                          BNE     3$              ;Br if yes
004104  000137  001500'                 JMP     NEWLIN          ;Ignore if no cell open
004110  105737  000000G         3$:     TSTB    D.V1FL          ;Was a new value specified for this cell?
004114  003404                          BLE     1$              ;Br if not
004116  013700  000000G                 MOV     D.VAL1,R0       ;Get new value
004122  004737  006450'                 CALL    STRVAL          ;Store into currently open cell
004126  105737  000000G         1$:     TSTB    D.BYTM          ;Are we in byte or word mode?
004132  001017                          BNE     2$              ;Br if in byte mode
004134  162737  000002  000000G         SUB     #2,D.LOC        ;Move to previous word
004142                                  TPRINT  #CRLF           ;Go to new line
004152  004737  007100'                 CALL    SHOLOC          ;Display address of new location
004156                                  PRINT   </>             ;Display "/"
004166  000137  001656'                 JMP     CMDSL1          ;Simulate "/" command
004172  005337  000000G         2$:     DEC     D.LOC           ;Move to previous byte
004176                                  TPRINT  #CRLF           ;go to new line
004206  004737  007100'                 CALL    SHOLOC          ;Display address of new location
004212                                  PRINT   <\>             ;Display "\"
004222  000137  001734'                 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.
                                ;
004226  010046                  DBGINI: MOV     R0,-(SP)
                                ;
                                ;  Initialize some debugger values
                                ;
004230  012700  000000G                 MOV     #D.START,R0     ;Point to start of debugging data
004234  005020                  1$:     CLR     (R0)+           ;Clear all debugging data to 0
004236  020027  000000G                 CMP     R0,#D.END       ;Finished?
004242  103774                         BLO     1$              ;Loop if not
004244  012737  177777  000000G         MOV     #177777,D.MASK  ;Initialize the data mask
004252  052737  000000G 000000G         BIS     #D$INIT,D.FLAG  ;Say initialization has been done
                                ;
                                ;  Set previous-mode = user in our PSW so we can use MFPD/MTPD to
                                ;  access user's program.
                                ;
004260  052737  000000G 000000G         BIS     #UPMODE,@#PSW   ;Say previous mode = user
                                ;
                                ;  Print greeting message
                                ;
004266                                  TPRINT  #TM$GRT         ;Print greeting message
                                ;
                                ;  Finished
                                ;
004276  012600                          MOV     (SP)+,R0
004300  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.
                                ;
004302  010146                  BRKRST: MOV     R1,-(SP)
004304  032737  000000G 000000G         BIT     #D$BKST,D.FLAG  ;Are breakpoints set in program?
004312  001427                          BEQ     9$              ;Br if not
004314  032737  000000G 000000G         BIT     #D$CKBK,D.FLAG  ;Are there any active instruction breaks?
004322  001423                          BEQ     9$              ;Br if not
                                ;
                                ;  Reset instructions that were overwritten by breakpoints
                                ;
004324  005001                          CLR     R1              ;Init index into breakpoint tables
004326  016100  000000G         1$:     MOV     D.BKAD(R1),R0   ;Get address of breakpoint
004332  001407                          BEQ     2$              ;Br if this breakpoint not in use
004334  006510                          MFPI    (R0)            ;Get breakpoint instruction
004336  022627  000003                  CMP     (SP)+,#3        ;Is this a breakpoint instruction?
004342  001003                          BNE     2$              ;Br if not
004344  016146  000000G                 MOV     D.BKSV(R1),-(SP);Get saved instruction
004350  006610                          MTPI    (R0)            ;Restore saved instruction
004352  062701  000002          2$:     ADD     #2,R1           ;Advance breakpoint index
004356  020127  000020                  CMP     R1,#16.         ;Checked all breakpoints?
004362  101761                          BLOS    1$              ;Loop if more to check
004364  042737  000000G 000000G         BIC     #D$BKST,D.FLAG  ;Say breakpoints no longer set
                                ;
                                ;  Finished
                                ;
004372  012601                  9$:     MOV     (SP)+,R1
004374  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.
                                ;
004376  010146                  BRKSET: MOV     R1,-(SP)
004400  032737  000000G 000000G         BIT     #D$CKBK,D.FLAG  ;Are there any instruction breakpoints?
004406  001416                          BEQ     3$              ;Br if not
                                ;
                                ;  Set breakpoints
                                ;
004410  012701  000020                  MOV     #16.,R1         ;Get index for last breakpoint
004414  016100  000000G         1$:     MOV     D.BKAD(R1),R0   ;Get address where breakpoint goes
004420  001406                          BEQ     2$              ;Br if this breakpoint not in use
004422  006510                          MFPI    (R0)            ;Get current contents of location
004424  012661  000000G                 MOV     (SP)+,D.BKSV(R1);Save original contents of cell
004430  012746  000003                  MOV     #3,-(SP)        ;Push BPT instruction
004434  006610                          MTPI    (R0)            ;Store BPT instruction into user's program
004436  162701  000002          2$:     SUB     #2,R1           ;More breakpoints to check?
004442  002364                          BGE     1$              ;Br if yes
004444  052737  000000G 000000G 3$:     BIS     #D$BKST,D.FLAG  ;Set flag saying breakpoints set in program
                                ;
                                ;  Finished
                                ;
004452  012601                          MOV     (SP)+,R1
004454  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
                                ;
004456  010146                  BRKCHK: MOV     R1,-(SP)
                                ;
                                ;  Clear instruction and data breakpoint flags
                                ;
004460  042737  000000C 000000G         BIC     #<D$DBRK!D$IBRK!D$SBRK>,D.FLAG ;No breakpoints yet
                                ;
                                ;  See if a data breakpoint is active
                                ;
004466  032737  000000G 000000G         BIT     #D$DMON,D.FLAG  ;Is a data breakpoint active?
004474  001425                        BEQ     1$              ;Br if not
004476  013701  000000G               MOV     D.DADR,R1       ;Get address of monitored cell
004502  106511                        MFPD    (R1)            ;Get contents of monitored cell
004504  013701  000000G               MOV     D.MASK,R1       ;Get data mask
004510  005101                        COM     R1              ;Complement
004512  040116                        BIC     R1,(SP)         ;Leave only bits of interest
004514  032737  000000G 000000G       BIT     #D$DVAL,D.FLAG  ;Are we waiting for a specified value?
004522  001004                        BNE     2$              ;Br if yes
                                ;  Trigger data breakpoint whenever monitored data cell changes value
004524  022637  000000G               CMP     (SP)+,D.DOLD    ;Has value changed?
004530  001004                        BNE     3$              ;Br if yes -- trigger breakpoint
004532  000406                        BR      1$              ;Hasn't changed -- no data break
                                ;  Trigger data breakpoint if value takes on specified value
004534  022637  000000G         2$:   CMP     (SP)+,D.DTRG    ;Does cell have value of interest?
004540  001003                        BNE     1$              ;Br if not -- no data breakpoint
                                ;
                                ;  Trigger a data breakpoint
                                ;
004542  052737  000000G 000000G 3$:   BIS     #D$DBRK,D.FLAG  ;Set data-breakpoint flag
                                ;
                                ;  See if we need to check for instruction or single-step breakpoints
                                ;
004550  032737  000000C 000000G 1$:   BIT     #<D$SSTP!D$TSTP!D$CKBK>,D.FLAG ;Any instruction breaks or stp?
004556  001442                        BEQ     9$              ;Br if neither instruction brks or single step
                                ;
                                ;  Ignore single-step and instruction breakpoints if we just
                                ;  executed a pending instruction.
```

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

```
                                        .SBTTL  SHOBRK -- Display breakpoint information
                                ;-----------------------------------------------------------------
                                ;  SHOBRK is called to display information about a breakpoint
                                ;  that has been triggered.
                                ;
004670  010146                  SHOBRK: MOV     R1,-(SP)
004672                                  TPRINT  #CRLF           ;Get to start of new line
                                ;
                                ;  See if a data breakpoint occurred
                                ;
004702  032737  000000G 000000G         BIT     #D$DBRK,D.FLAG  ;Did a data breakpoint occur?
004710  001454                          BEQ     1$              ;Br if not
004712                                  PRINT   <Data break at >
004736  013700  000000G                 MOV     D.R7,R0         ;Get breakpoint PC
004742  004737  007276'                 CALL    LSTADR          ;Display the address
004746                                  PRINT   < -- >
004760  013700  000000G                 MOV     D.DADR,R0       ;Get address of monitored cell
004764  004737  007276'                 CALL    LSTADR          ;Display it
004770                                  PRINT   < = >
005000  013700  000000G                 MOV     D.DADR,R0       ;Get address of cell
005004  106510                          MFPD    (R0)            ;Get current contents of cell
005006  012600                          MOV     (SP)+,R0
005010  010037  000000G                 MOV     R0,D.DOLD       ;Save old cell value
005014  004737  006136'                 CALL    PRTWRD          ;Display value of cell
005020  013700  000000G                 MOV     D.MASK,R0       ;Get data mask
005024  005100                          COM     R0              ;Complement
005026  040037  000000G                 BIC     R0,D.DOLD       ;Clear all but bits of interest
005032                                  TPRINT  #CRLF           ;Go to new line
                                ;
                                ;  See if a single-step breakpoint occurred
                                ;
005042  032737  000000G 000000G 1$:     BIT     #D$SBRK,D.FLAG  ;Did a single-step breakpoint occur?
005050  001406                          BEQ     2$              ;Br if not
005052                          5$:     PRINT   <Step:>         ;Print heading
005064  000451                          BR      3$              ;Now treat like instruction breakpoing
                                ;
                                ;  See if an instruction breakpoint occurred
                                ;
005066  032737  000000G 000000G 2$:     BIT     #D$IBRK,D.FLAG  ;Instruction breakpoint?
005074  001426                          BEQ     4$              ;Br if not
005076                                  TPRINT  #CRLF           ;Go to new line
005106  123727  000000G 000010          CMPB    D.BKNM,#8.      ;CALL breakpoint?
005114  001003                          BNE     6$              ;Br if not
005116  005037  000000G                 CLR     D.CBRK          ;Clear the call breakpoint
005122  000753                          BR      5$              ;Br if yes -- Treat like single step break
005124                          6$:     PRINT   <B>
005132  113700  000000G                 MOVB    D.BKNM,R0       ;Get breakpoint number
005136  004737  006030'                 CALL    PRTOCT          ;Print breakpoint number
005142                                  PRINT   <;>
005150  000417                          BR      3$
                                ;
                                ;  See if we had a forced breakpoint
                                ;
005152  032737  000000G 000000G 4$:     BIT     #D$FBRK,D.FLAG  ;Forced breakpoint?
005160  001427                          BEQ     9$              ;Br if not
005162                                  TPRINT  #CRLF           ;Go to a new line
005172                                  PRINT   <Break at >
```

```
                                        ;
                                        ;  Display the address of the breakpoint
                                        ;
005210  013700  000000G         3$:     MOV     D.R7,R0         ;Get address where break occurred
005214  004737  007276'                 CALL    LSTADR          ;Display the address
                                        ;
                                        ;  Display the decoded instruction
                                        ;
005220  013701  000000G                 MOV     D.R7,R1         ;Get address of break
005224  004737  007640'                 CALL    DECODE          ;Display the decoded instruction
005230                                  TPRINT  #CRLF           ;Go to new line
                                        ;
                                        ;  Finished
                                        ;
005240  012601                  9$:     MOV     (SP)+,R1
005242  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.
                                ;
005244  010146                  FLGBRK: MOV     R1,-(SP)
                                ;
                                ;  Determine if there are any normal instruction breakpoints
                                ;
005246  012701  000020                  MOV     #16.,R1         ;Get index to last breakpoint entry
005252  005761  000000G         5$:     TST     D.BKAD(R1)      ;Is this breakpoint set?
005256  001007                          BNE     6$              ;Br if yes
005260  162701  000002                  SUB     #2,R1           ;More to check?
005264  002372                          BGE     5$              ;Loop if yes
                                ;
                                ;  No breakpoints are active
                                ;
005266  042737  000000G 000000G        BIC     #D$CKBK,D.FLAG  ;No breakpoints are set
005274  000403                          BR      9$
                                ;
                                ;  There are active breakpoints
                                ;
005276  052737  000000G 000000G 6$:     BIS     #D$CKBK,D.FLAG  ;Remember some breakpoint is set
                                ;
                                ;  Finished
                                ;
005304  012601                  9$:     MOV     (SP)+,R1
005306  000207                          RETURN
```

```
                        .SBTTL  ACRVAL -- Accrue a general value
                ;-----------------------------------------------------------------------
                ;  ACRVAL is called to accrue a general value of the form
                ;  [r,]value    or    $x
                ;  where "r" is a relocation base register number,
                ;  and "x" is an internal register number.
                ;
                ;  Outputs:
                ;   R0 = Value accrued.
                ;   R1 = Status flag:
                ;        +1 ==> normal value
                ;         0 ==> No value gotten.
                ;        -1 ==> Internal register address.
                ;
005310  010246  ACRVAL: MOV     R2,-(SP)
                ;
                ;  Get 1st character and see if this is a normal value or a register name.
                ;
005312  004737  005756'         CALL    GETCHR          ;Get 1st character
005316  120027  000044          CMPB    R0,#'$          ;Is this a register value?
005322  001461                  BEQ     1$              ;Branch if yes
005324  120027  000056          CMPB    R0,#'.          ;"." = Current program counter
005330  001451                  BEQ     10$             ;Br if period
005332  004737  006022'         CALL    PSHCHR          ;Save the character
005336  120027  000053          CMPB    R0,#'+          ;Is this a sign character?
005342  001414                  BEQ     2$              ;Br if yes
005344  120027  000055          CMPB    R0,#'-
005350  001411                  BEQ     2$
005352  120027  000060          CMPB    R0,#'0          ;Is this character a digit?
005356  103403                  BLO     8$              ;Br if not a digit
005360  120027  000071          CMPB    R0,#'9
005364  101403                  BLOS    2$              ;Br if a digit
                ;
                ;  No value was specified
                ;
005366  005001  8$:     CLR     R1              ;Return status code
005370  005000                  CLR     R0              ;Return value of 0
005372  000457                  BR      9$
                ;
                ;  Accrue normal value
                ;
005374  004737  005536'         2$:     CALL    ACRNUM          ;Accrue first part of number
                ;
                ;  See if 1st part of value is a relocation base register number
                ;
005400  010102                  MOV     R1,R2           ;Save value that was accrued
005402  120027  000054          CMPB    R0,#',          ;Was first part terminated with a comma?
005406  001016                  BNE     3$              ;Br if not
005410  020227  000007          CMP     R2,#7           ;Is first part a valid register number?
005414  101404                  BLOS    4$              ;Br if ok
005416                          ERR     #EM$IRB
005426  006302          4$:     ASL     R2              ;Convert reg # to word table index
005430  004737  005756'         CALL    GETCHR          ;Skip over comma
005434  004737  005536'         CALL    ACRNUM          ;Accrue second part of number
005440  066201  000000G         ADD     D.RLBS(R2),R1   ;Add relocation base to offset value
005444  010100          3$:     MOV     R1,R0           ;Return value in R0
005446  012701  000001          MOV     #1,R1           ;Return status code in R1
```

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

```
                                        .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.
                                ;
005536  010246                  ACRNUM: MOV     R2,-(SP)
005540  010346                          MOV     R3,-(SP)
005542  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if number has a leading sign character
                                ;
005544  005002                          CLR     R2              ;Assume result should be positive
005546  004737  005756'                 CALL    GETCHR          ;Get 1st character of number
005552  120027  000053                  CMPB    R0,#'+          ;Leading plus sign?
005556  001407                          BEQ     6$              ;Br if yes
005560  120027  000055                  CMPB    R0,#'-          ;Leading minus sign?
005564  001002                          BNE     8$              ;Br if not
005566  005202                          INC     R2              ;Set flag to negate the value
005570  000402                          BR      6$
005572  004737  006022'          8$:     CALL    PSHCHR          ;Push the first digit
                                ;
                                ;  Scan the number and store in D.NMBF
                                ;
005576  012701  000000G          6$:     MOV     #D.NMBF,R1      ;Point to number character buffer
005602  012705  000010                  MOV     #8.,R5          ;Assume this is an octal value
005606  004737  005756'          1$:     CALL    GETCHR          ;Get next character of number
005612  120027  000060                  CMPB    R0,#'0          ;Is this character a digit?
005616  103415                          BLO     2$              ;Br if not
005620  120027  000071                  CMPB    R0,#'9          ;Decimal digit?
005624  101012                          BHI     2$              ;Br if not
005626  120027  000067                  CMPB    R0,#'7          ;Octal digit?
005632  101402                          BLOS    3$              ;Br if yes
005634  012705  000012                  MOV     #10.,R5         ;This must be a decimal value
005640  020127  000000G          3$:     CMP     R1,#D.NMBE      ;Are we about to overflow the buffer?
005644  103040                          BHIS    11$             ;Br if yes
005646  110021                          MOVB    R0,(R1)+        ;Store character into buffer
005650  000756                          BR      1$              ;Loop to get rest of number
                                ;
                                ;  Hit end of number
                                ;
005652  105011                  2$:     CLRB    (R1)            ;Put null at end of buffer
005654  120027  000056                  CMPB    R0,#'.          ;Decimal point at end of number?
005660  001003                          BNE     4$              ;Br if not
005662  012705  000012                  MOV     #10.,R5         ;Remember this is a decimal value
005666  000402                          BR      5$
005670  004737  006022'          4$:     CALL    PSHCHR          ;Save delimiter
                                ;
                                ;  Convert number string to binary value
                                ;
005674  012703  000000G          5$:     MOV     #D.NMBF,R3      ;Point to start of buffer
005700  005001                          CLR     R1              ;Store value in R1
005702  112300                  7$:     MOVB    (R3)+,R0        ;Get next character
005704  001405                          BEQ     9$              ;Br if hit end of number
```

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

```
                                        .SBTTL  GETCHR -- Get next character from terminal
                                ;-----------------------------------------------------------------------
                                ;  Get the next character from the terminal.
                                ;
                                ;  Outputs:
                                ;   R0 = Character gotten
                                ;
005756                          GETCHR:
                                ;
                                ;  See if PSHCHR was called to save a character
                                ;
005756  113700  000000G                 MOVB    D.SVCH,R0       ;Has a character been pushed?
005762  001014                          BNE     9$              ;Br if yes
                                ;
                                ;  No saved character.
                                ;  Get character from terminal.
                                ;
005764                                  .TTYIN                  ;Get character from terminal
005770  005237  000000G                 INC     D.PCOL          ;Advance print column
                                ;
                                ;  Convert lower-case characters to upper-case
                                ;
005774  120027  000141                  CMPB    R0,#141         ;Lower-case a
006000  103405                          BLO     9$
006002  120027  000172                  CMPB    R0,#172         ;Lower-case z
006006  101002                          BHI     9$
006010  162700  000040                  SUB     #40,R0          ;Convert lower-case to upper-case
                                ;
                                ;  Finished
                                ;
006014  105037  000000G         9$:     CLRB    D.SVCH          ;No saved character
006020  000207                          RETURN
                                
                                        .SBTTL  PSHCHR -- Push a character
                                ;-----------------------------------------------------------------------
                                ;  Push a character so GETCHR will get it on next call.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be pushed.
                                ;
006022  110037  000000G         PSHCHR: MOVB    R0,D.SVCH       ;Save the character for GETCHR
006026  000207                          RETURN                  ;Finished
```

```
                                        .SBTTL  PRTOCT -- Print octal value
                                ;-------------------------------------------------------------------
                                ;  Print an octal value with no leading zeroes.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to print.
                                ;
006030  010146          PRTOCT: MOV     R1,-(SP)
006032  010246                  MOV     R2,-(SP)
006034  010346                  MOV     R3,-(SP)
006036  005003                  CLR     R3              ;Say non-zero digit not seen yet
006040  010001                  MOV     R0,R1           ;Get value to print
006042  005000                  CLR     R0              ;Get 1st bit into R0
006044  073027  000001          ASHC    #1,R0
006050  012702  000006          MOV     #6,R2           ;Print total of 6 digits
006054  000403                  BR      2$
006056  005000          1$:     CLR     R0              ;Get next digit into R0
006060  073027  000003          ASHC    #3,R0
006064  050003          2$:     BIS     R0,R3           ;Remember if non-zero digit seen
006066  001406                  BEQ     3$              ;Br if no non-zero digits seen yet
006070  062700  000060          ADD     #'0,R0          ;Convert to ascii character
006074                          .TTYOUT                 ;Print the character
006100  005237  000000G         INC     D.PCOL          ;Advance print column
006104  077214          3$:     SOB     R2,1$           ;Loop if more digits to print
006106  005703                  TST     R3              ;Did we see any non-zero digits?
006110  001006                  BNE     4$              ;Br if yes
006112                          .TTYOUT #'0             ;Print one zero if not
006122  005237  000000G         INC     D.PCOL          ;Advance print column
006126  012603          4$:     MOV     (SP)+,R3
006130  012602                  MOV     (SP)+,R2
006132  012601                  MOV     (SP)+,R1
006134  000207                  RETURN
```

```
                                        .SBTTL  PRTWRD -- Print octal word value
                                ;-------------------------------------------------------------
                                ;  Print a 16-bit octal value;
                                ;  Inputs:
                                ;   R0 = Value to be printed.
                                ;
006136  010146          PRTWRD: MOV     R1,-(SP)
006140  010246                  MOV     R2,-(SP)
006142  010001                  MOV     R0,R1           ;Get value to print
006144  005000                  CLR     R0              ;Get 1st bit into R0
006146  073027  000001          ASHC    #1,R0
006152  012702  000006          MOV     #6,R2           ;Print total of 6 digits
006156  000403                  BR      2$
006160  005000          1$:     CLR     R0              ;Get next digit into R0
006162  073027  000003          ASHC    #3,R0
006166  062700  000060          ADD     #'0,R0          ;Convert to ascii character
006172                          .TTYOUT                 ;Print the character
006176  005237  000000G         INC     D.PCOL          ;Advance print column
006202  077212                  SOB     R2,1$           ;Loop if more digits to print
006204  012602                  MOV     (SP)+,R2
006206  012601                  MOV     (SP)+,R1
006210  000207                  RETURN

                                        .SBTTL  PRTBYT -- Print octal byte value
                                ;-------------------------------------------------------------
                                ;  Print an 8-bit octal value.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to be printed.
                                ;
006212  010146          PRTBYT: MOV     R1,-(SP)
006214  010246                  MOV     R2,-(SP)
006216  010001                  MOV     R0,R1           ;Get value to be printed
006220  000301                  SWAB    R1              ;Left-justify value in R1
006222  005000                  CLR     R0              ;Move 1st 2 bits into R0
006224  073027  000002          ASHC    #2,R0
006230  012702  000003          MOV     #3,R2           ;Print 3 digits total
006234  000403                  BR      2$
006236  005000          1$:     CLR     R0              ;Get next octal digit to R0
006240  073027  000003          ASHC    #3,R0
006244  062700  000060          ADD     #'0,R0          ;Convert to ascii character
006250                          .TTYOUT                 ;Print the character
006254  005237  000000G         INC     D.PCOL          ;Advance print column
006260  077212                  SOB     R2,1$           ;Loop if more digits to print
006262  012602                  MOV     (SP)+,R2
006264  012601                  MOV     (SP)+,R1
006266  000207                  RETURN
```

```
                                        .SBTTL  PRTR50 -- Print a RAD50 value
                                ;-------------------------------------------------------------------
                                ;  PRTR50 is called to print a RAD50 (radix 50) value.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to be printed.
                                ;
006270  010146                  PRTR50: MOV     R1,-(SP)
006272  010246                          MOV     R2,-(SP)
                                ;
                                ;  Convert value to ascii character string and stack the characters
                                ;
006274  012702  000003                  MOV     #3,R2           ;Get # chars to convert
006300  005046                          CLR     -(SP)           ;Put null on stack to signal end
006302  010001                          MOV     R0,R1           ;Get value to be converted
006304  005000                  1$:     CLR     R0              ;Clear high-order for divide
006306  071027  000050                  DIV     #50,R0          ;Divide R0-R1 by 50
006312  116146  000220'                MOVB    R50CHR(R1),-(SP);Convert remainder to ascii and stack it
006316  010001                          MOV     R0,R1           ;Get quotient
006320  077207                          SOB     R2,1$           ;Loop if more to convert
                                ;
                                ;  Finished conversion.  Print the result.
                                ;
006322  012600                  2$:     MOV     (SP)+,R0        ;Get char to print
006324  001405                          BEQ     3$              ;Br if hit end of string
006326                                  .TTYOUT                 ;Print the character
006332  005237  000000G                 INC     @#D.PCOL        ;Advance print column
006336  000771                          BR      2$              ;Loop to print more
                                ;
                                ;  Finished
                                ;
006340  012602                  3$:     MOV     (SP)+,R2
006342  012601                          MOV     (SP)+,R1
006344  000207                          RETURN
```

```
                                        .SBTTL  ERRPRT -- Print an error message
                                ;-------------------------------------------------------------
                                ;  Print the error message whose address is in R1 and then go and
                                ;  get a new command.
                                ;
006346                          ERRPRT: TPRINT  #CRLF           ;Go to new line
006356                                  TPRINT  R1              ;Print error message
006364  005037  000000G                 CLR     D.PCOL          ;Say print column = 0
006370  013706  000000G                 MOV     D.SPSV,SP       ;Reset stack pointer
006374  000137  001510'                 JMP     NEWCMD          ;Go get a new command

                                        .SBTTL  LSTTXT -- Print text string
                                ;-------------------------------------------------------------
                                ;  LSTTXT is called by use of the PRINT macro.  It prints a text
                                ;  string on the terminal without carriage-return, line-feed.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of string in ASCIZ form that follows call.
                                ;
006400  112200                  LSTTXT: MOVB    (R2)+,R0        ;Get next char from text string
006402  001405                          BEQ     2$              ;Br if hit end of string
006404                                  .TTYOUT                 ;Print the character
006410  005237  000000G                 INC     D.PCOL          ;Advance print column
006414  000771                          BR      LSTTXT          ;Loop till end of string hit
006416  005202                  2$:     INC     R2              ;Bound up to even byte address
006420  042702  000001                  BIC     #1,R2
006424  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 006426  113700  0000000          SETLOC: MOVB    D.V1FL,R0       ;Get mode of address
     14 006432  001405                           BEQ     9$              ;Br if no address specified
     15 006434  110037  0000000                  MOVB    R0,D.LOCM       ;Set mode of current cell
     16 006440  013737  0000000 0000000          MOV     D.VAL1,D.LOC    ;Set address of current cell
     17 006446  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.
                ;
006450  010146          STRVAL: MOV     R1,-(SP)
006452  010246                  MOV     R2,-(SP)
                ;
                ;  Make sure a cell is open
                ;
006454  105737  000000G         TSTB    D.LOCM          ;Is a cell open now?
006460  001512                  BEQ     9$              ;Br if not
                ;
                ;  Determine if we are storing into user's job or internal cell
                ;
006462  105737  000000G         TSTB    D.LOCM          ;Internal or user job?
006466  100457                  BMI     1$              ;Br if internal cell
                ;
                ;  Store into user's job
                ;
006470  105737  000000G         TSTB    D.BYTM          ;Byte or word mode?
006474  001013                  BNE     2$              ;Br if byte mode
                ;
                ;  Store into word
                ;
006476  010046                  MOV     R0,-(SP)        ;Put value on stack
006500  013700  000000G         MOV     D.LOC,R0        ;Get address where we should store
006504  032737  000000G 000000G BIT     #D$ISPC,D.FLAG  ;Should we use I-space?
006512  001402                  BEQ     4$              ;Br if not (D-space is default)
006514  006610                  MTPI    (R0)            ;Store into program I-space
006516  000473                  BR      9$
006520  106610          4$:     MTPD    (R0)            ;Store into user's area
006522  000471                  BR      9$
                ;
                ;  Store into byte
                ;
006524  042700  177400  2$:     BIC     #^C377,R0       ;Clear high-order byte of value being stored
006530  013702  000000G         MOV     D.LOC,R2        ;Get address where we are to store
006534  042702  000001          BIC     #1,R2           ;Force address to be even
006540  032737  000000G 000000G BIT     #D$ISPC,D.FLAG  ;I-space reference?
006546  001402                  BEQ     7$              ;Br if not
006550  006512                  MFPI    (R2)            ;Get current I-space value
006552  000401                  BR      8$              ;And continue
006554  106512          7$:     MFPD    (R2)            ;Get current contents of word (D-space)
006556  032737  000001  000000G 8$:     BIT     #1,D.LOC        ;Even or odd byte?
006564  001004                  BNE     3$              ;Br if odd byte
006566  042716  000377          BIC     #377,(SP)       ;Clear low-order byte of old value
006572  050016                  BIS     R0,(SP)         ;Put in new low-order byte
006574  000404                  BR      5$              ;Go set into user space
006576  042716  177400  3$:     BIC     #177400,(SP)    ;Clear high-order byte
006602  000300                  SWAB    R0              ;Get new value to high-order byte
006604  050016                  BIS     R0,(SP)         ;Store new high-order byte
```

```
     58 006606  032737  000000G 000000G 5$:     BIT     #D$ISPC,D.FLAG  ;Store into I-space?
     59 006614  001402                         BEQ     6$              ;Br if not (D-space is default)
     60 006616  006612                         MTPI    (R2)            ;Store into user I-space
     61 006620  000432                         BR      9$
     62 006622  106612                 6$:     MTPD    (R2)            ;Store into user D-space
     63 006624  000430                         BR      9$
     64                                 ;
     65                                 ;  Store into internal register
     66                                 ;
     67 006626  105737  000000G         1$:     TSTB    D.BYTM          ;Byte or word mode?
     68 006632  001003                         BNE     12$             ;Br if byte mode
     69                                 ;
     70                                 ;  Store into word
     71                                 ;
     72 006634  010077  000000G                 MOV     R0,@D.LOC       ;Store into internal cell
     73 006640  000422                         BR      9$
     74                                 ;
     75                                 ;  Store into byte
     76                                 ;
     77 006642  042700  177400          12$:    BIC     #^C377,R0       ;Clear high-order byte of value being stored
     78 006646  013702  000000G                 MOV     D.LOC,R2        ;Get address where we are to store
     79 006652  042702  000001                 BIC     #1,R2           ;Force address to be even
     80 006656  032737  000001  000000G         BIT     #1,D.LOC        ;Even or odd byte?
     81 006664  001344                         BNE     3$              ;Br if odd byte
     82 006666  042712  000377                 BIC     #377,(R2)       ;Clear low-order byte of old value
     83 006672  050012                         BIS     R0,(R2)         ;Put in new low-order byte
     84 006674  000404                         BR      9$
     85 006676  042712  177400          13$:    BIC     #177400,(R2)    ;Clear high-order byte
     86 006702  000300                         SWAB    R0              ;Get new value to high-order byte
     87 006704  050012                         BIS     R0,(R2)         ;Store new high-order byte
     88                                 ;
     89                                 ;  Finished
     90                                 ;
     91 006706  012602                 9$:     MOV     (SP)+,R2
     92 006710  012601                         MOV     (SP)+,R1
     93 006712  000207                         RETURN
```

```
                                        .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
                                ;
006714  010246                  GETVAL: MOV     R2,-(SP)
                                ;
                                ;  Determine if we are fetching from user's job or internal cell
                                ;
006716  105737  000000G                 TSTB    D.LOCM          ;Internal or user job?
006722  100442                          BMI     1$              ;Br if internal cell
                                ;
                                ;  Fetch from user's job
                                ;
006724  105737  000000G                 TSTB    D.BYTM          ;Byte or word mode?
006730  001013                          BNE     2$              ;Br if byte mode
                                ;
                                ;  Fetch from word
                                ;
006732  013700  000000G                 MOV     D.LOC,R0        ;Get address of value wanted
006736  032737  000000G 000000G         BIT     #D$ISPC,D.FLAG  ;Should we use I-space?
006744  001402                          BEQ     4$              ;Br if not (D-space default)
006746  006510                          MFPI    (R0)            ;Get value from user I-space
006750  000401                          BR      5$              ;And continue
006752  106510                  4$:     MFPD    (R0)            ;Get value from user D-space
006754  012600                  5$:     MOV     (SP)+,R0        ;Return in R0
006756  000446                          BR      9$
                                ;
                                ;  Fetch from byte
                                ;
006760  013702  000000G         2$:     MOV     D.LOC,R2        ;Get address of word
006764  042702  000001                  BIC     #1,R2           ;Force address to be even
006770  032737  000000G 000000G         BIT     #D$ISPC,D.FLAG  ;Should we use I-space
006776  001402                          BEQ     6$              ;Br if not (D-space default)
007000  006512                          MFPI    (R2)            ;Get current value from I-space
007002  000401                          BR      7$              ;And continue
007004  106512                  6$:     MFPD    (R2)            ;Get current value from D-space
007006  012600                  7$:     MOV     (SP)+,R0        ;Get word value to R0
007010  032737  000001  000000G         BIT     #1,D.LOC        ;Even or odd byte?
007016  001401                          BEQ     3$              ;Br if low-order byte wanted
007020  000300                          SWAB    R0              ;Swap bytes if high-order byte wanted
007022  042700  177400          3$:     BIC     #177400,R0      ;Clear high-order byte
007026  000422                          BR      9$
                                ;
                                ;  Fetch from internal register
                                ;
007030  105737  000000G         1$:     TSTB    D.BYTM          ;Byte or word mode?
007034  001003                          BNE     12$             ;Br if byte mode
                                ;
                                ;  Fetch from word
```

```
                                        ;
007036  017700  000000G                 MOV     @D.LOC,R0       ;Get the value
007042  000414                          BR      9$
                                        ;
                                        ;  Fetch from byte
                                        ;
007044  013702  000000G         12$:    MOV     D.LOC,R2        ;Get address where we are to fetch
007050  042702  000001                  BIC     #1,R2           ;Force address to be even
007054  011200                          MOV     (R2),R0         ;Get the word value
007056  032737  000001  000000G         BIT     #1,D.LOC        ;Even or odd byte?
007064  001401                          BEQ     13$             ;Br if even byte wanted
007066  000300                          SWAB    R0              ;Swap high-order byte to low-order
007070  042700  177400          13$:    BIC     #177400,R0      ;Clear high-order byte
                                        ;
                                        ;  Finished
                                        ;
007074  012602                  9$:     MOV     (SP)+,R2
007076  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 ($x).
                                ;
                                ;  Inputs:
                                ;   D.LOC  = Address to be displayed
                                ;   D.LOCM = Mode of address (+1==>In user's program, -1==>Internal cell)
                                ;
007100  010146                  SHOLOC: MOV     R1,-(SP)
007102  010246                          MOV     R2,-(SP)
007104  010346                          MOV     R3,-(SP)
007106  010446                          MOV     R4,-(SP)
                                ;
                                ;  Determine if this is an internal or external cell
                                ;
007110  105737  000000G                 TSTB    D.LOCM          ;Internal or external address?
007114  002405                          BLT     1$              ;Br if internal address
                                ;
                                ;  Display address of cell in user's program
                                ;
007116  013700  000000G                 MOV     D.LOC,R0        ;Get address to be displayed
007122  004737  007276'                 CALL    LSTADR          ;Display relative to relocation base
007126  000456                          BR      9$
                                ;
                                ;  Display name of internal cell
                                ;
007130  013700  000000G         1$:     MOV     D.LOC,R0        ;Get address of internal cell
007134  012701  000402'                 MOV     #INTADR,R1      ;Point to table of addresses of cells
007140  012702  177777                  MOV     #-1,R2          ;Init for search
007144  005003                          CLR     R3
007146  020011                  2$:     CMP     R0,(R1)         ;Is address below base of this cell?
007150  103406                          BLO     3$              ;Br if yes
007152  010004                          MOV     R0,R4           ;Get address of interest
007154  161104                          SUB     (R1),R4         ;Determine distance from cell base addr
007156  020402                          CMP     R4,R2           ;Is this the closest we've seen so far?
007160  101002                          BHI     3$              ;Br if not
007162  010402                          MOV     R4,R2           ;Remember offset from nearest cell
007164  010103                          MOV     R1,R3           ;Remember address of nearest cell
007166  062701  000002          3$:     ADD     #2,R1           ;Point to address of next cell
007172  020127  000434'                 CMP     R1,#INTEND      ;Checked all cells?
007176  103763                          BLO     2$              ;Loop if not
                                ;
                                ;  Display register name
                                ;
007200  162703  000402'                 SUB     #INTADR,R3      ;Get offset into address table
007204  006203                          ASR     R3              ;Convert to byte table index
007206                                  .TTYOUT #'$             ;Print "$"
007216  005237  000000G                 INC     D.PCOL          ;Advance print column
007222  116300  000364'                 MOVB    INTCHR(R3),R0   ;Get name of internal register
007226                                  .TTYOUT                 ;Print register name
007232  005237  000000G                 INC     D.PCOL          ;Advance print column
                                ;
                                ;  Display offset from register
                                ;
007236  005702                          TST     R2              ;Any offset from base of cell?
```

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

```
                                        .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.
                                ;
007276                          LSTADR:
                                ;
                                ;  See if DP$LAA is set in $F register
                                ;
007276  032737  000000G 000000G         BIT     #DP$LAA,D.PFMT  ;Absolute addresses wanted?
007304  001003                          BNE     1$              ;Br if yes
                                ;
                                ;  Display address with relocation base
                                ;
007306  004737  007322'                 CALL    SHOADR          ;Display address with relocation base
007312  000402                          BR      9$
                                ;
                                ;  Display absolute address
                                ;
007314  004737  006030'         1$:     CALL    PRTOCT          ;Display absolute address
                                ;
                                ;  Finished
                                ;
007320  000207                  9$:     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.
                                ;
007322  010146                  SHOADR: MOV     R1,-(SP)
007324  010246                          MOV     R2,-(SP)
007326  010346                          MOV     R3,-(SP)
007330  010446                          MOV     R4,-(SP)
                                ;
                                ;  Determine if address is relative to a relocation register
                                ;
007332  010004                          MOV     R0,R4           ;Save original address
007334  004737  007412'                  CALL    RELADR          ;Determine if addr is relative to reloc regn
007340  103414                          BCS     4$              ;Br if not
                                ;
                                ;  Print relocation register number
                                ;
007342  166304  000000G                 SUB     D.RLBS(R3),R4   ;Compute offset within the region
007346  010300                          MOV     R3,R0           ;Get register index
007350  006200                          ASR     R0              ;Convert index to number
007352  004737  006030'                  CALL    PRTOCT          ;Print register number
007356                                  .TTYOUT #54             ;Print ","
007366  005237  000000G                 INC     D.PCOL          ;Advance print column
                                ;
                                ;  Print offset within region
                                ;
007372  010400                  4$:     MOV     R4,R0           ;Get offset within region
007374  004737  006030'                  CALL    PRTOCT          ;Print the value
                                ;
                                ;  Finished
                                ;
007400  012604                          MOV     (SP)+,R4
007402  012603                          MOV     (SP)+,R3
007404  012602                          MOV     (SP)+,R2
007406  012601                          MOV     (SP)+,R1
007410  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.
                                ;
007412  010146                  RELADR: MOV     R1,-(SP)
007414  010246                          MOV     R2,-(SP)
007416  010446                          MOV     R4,-(SP)
                                ;
                                ;  Search for relocation region defined below the address
                                ;
007420  005001                          CLR     R1              ;Init relocation register index
007422  012702  177777                  MOV     #-1,R2          ;Init closest offset value
007426  010203                          MOV     R2,R3           ;Say no relocation base found yet
007430  005761  000000G         2$:     TST     D.RLBS(R1)      ;Is this relocation register in use?
007434  001412                          BEQ     3$              ;Br if not
007436  020061  000000G                 CMP     R0,D.RLBS(R1)   ;Is address above this relocation base?
007442  103407                          BLO     3$              ;Br if not
007444  010004                          MOV     R0,R4           ;Get address of interest
007446  166104  000000G                 SUB     D.RLBS(R1),R4   ;Calculate offset within region
007452  020402                          CMP     R4,R2           ;Is this the closest one so far?
007454  101002                          BHI     3$              ;Br if not
007456  010402                          MOV     R4,R2           ;Rember smallest offset
007460  010103                          MOV     R1,R3           ;Remember relocation register index
007462  062701  000002          3$:     ADD     #2,R1           ;Advance relocation register index
007466  020127  000016                  CMP     R1,#14.         ;Checked all relocation registers?
007472  101756                          BLOS    2$              ;Br if not
                                ;
                                ;  See if we found a relocation region
                                ;
007474  005703                          TST     R3              ;Did we find a relocation region
007476  002002                          BGE     5$              ;Br if yes
007500  000261                          SEC                     ;Say no region found
007502  000401                          BR      9$
007504  000241                  5$:     CLC                     ;Say a region was found
                                ;
                                ;  Finished
                                ;
007506  012604                  9$:     MOV     (SP)+,R4
007510  012602                          MOV     (SP)+,R2
007512  012601                          MOV     (SP)+,R1
007514  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.
                                ;
007516  010146                  DOPRT:  MOV     R1,-(SP)
007520  010246                          MOV     R2,-(SP)
                                ;
                                ;  Set up pointer to start of string
                                ;
007522  010002                          MOV     R0,R2           ;Get pointer to start of string
007524  113701  000000G                 MOVB    CORUSR,R1       ;Get current job index number
                                ;
                                ;  Get each char out of the string and print it
                                ;
007530  112200                  1$:     MOVB    (R2)+,R0        ;Get next char from string
007532  001411                          BEQ     2$              ;Br if hit null at end
007534  120027  000200                  CMPB    R0,#200         ;Is string terminated with a 200?
007540  001422                          BEQ     9$              ;Br if yes
007542                                  OCALL   PUTCHR          ;Print the character
007550  005237  000000G                 INC     D.PCOL          ;Advance print column
007554  000765                          BR      1$              ;Loop to print rest of string
                                ;
                                ;  Print Carriage-return Line-feed at end of string
                                ;
007556  012700  000015          2$:     MOV     #CR,R0          ;Get carriage return
007562                                  OCALL   PUTCHR          ;Print it
007570  012700  000012                  MOV     #LF,R0          ;Get line feed
007574                                  OCALL   PUTCHR          ;Print it
007602  005037  000000G                 CLR     D.PCOL          ;Say we are back to column 1
                                ;
                                ;  Finished
                                ;
007606  012602                  9$:     MOV     (SP)+,R2
007610  012601                          MOV     (SP)+,R1
007612  000207                          RETURN
```

```
                                        .SBTTL  DBGON  -- Place terminal in debug mode
                                ;-------------------------------------------------------------------
                                ;  DBGON is called to place the terminal in debug mode.
                                ;
007614  012700  000002'         DBGON:  MOV     #ONEMT,R0
007620  104375                          EMT     375
007622  000207                          RETURN
                                
                                        .SBTTL  DBGOFF -- Turn off terminal debug mode
                                ;-------------------------------------------------------------------
                                ;  DBGOFF is called to take the terminal out of debug mode
                                ;
007624  012700  000004'         DBGOFF: MOV     #OFFEMT,R0
007630  104375                          EMT     375
007632  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
                                ;
007634  000241                  CHKADR: CLC
007636  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.
                                ;
007640  010146                  DECODE: MOV     R1,-(SP)
007642  010246                          MOV     R2,-(SP)
007644  010346                          MOV     R3,-(SP)
007646  010446                          MOV     R4,-(SP)
007650  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize instruction length to 2 bytes
                                ;
007652  012737  000002  000000G         MOV     #2,D.ILEN       ;Say base instruction length = 2 bytes
                                ;
                                ;  Initialize column counter for decoded instruction display
                                ;
007660                                  PRINT   <    >          ;First print a few spaces
007672  005037  000000G                 CLR     D.PCOL          ;Say we are at column 1 now
                                ;
                                ;  Fetch the instruction word
                                ;
007676  006521                          MFPI    (R1)+           ;Get the instruction from user's space
007700  012602                          MOV     (SP)+,R2        ;Get instruction off of stack
                                ;
                                ;  Search the instruction list for this instruction
                                ;
007702  012704  012114'                 MOV     #INSBAS,R4      ;Point to base of instruction decode table
007706  010203                  1$:     MOV     R2,R3           ;Get instruction to be decoded
007710  046403  000000                  BIC     IB$MSK(R4),R3   ;Clear operand fields in instruction
007714  020364  000002                  CMP     R3,IB$VAL(R4)   ;Is this the instruction?
007720  001411                          BEQ     5$              ;Br if found the instruction
007722  116400  000005                  MOVB    IB$LEN(R4),R0   ;Get length of ascii name string
007726  005200                          INC     R0              ;Bound up to word boundary
007730  042700  000001                  BIC     #1,R0
007734  062700  000006                  ADD     #IB$NAM,R0      ;Add size of base portion of block
007740  060004                          ADD     R0,R4           ;Point to next instruction decode block
007742  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.
                                ;
007744  116403  000005          5$:     MOVB    IB$LEN(R4),R3   ;Get length of instruction mnemonic
007750  060337  000000G                 ADD     R3,D.PCOL       ;Advance column counter
007754  010405                          MOV     R4,R5           ;Point to the mnemonic ascii string
007756  062705  000006                  ADD     #IB$NAM,R5
007762  112500                  6$:     MOVB    (R5)+,R0        ;Get next char of mnemonic
007764                                  .TTYOUT                 ;Print the character
007770  077304                          SOB     R3,6$           ;Loop if more chars to print
                                ;
```

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

```
                                        .SBTTL  ODCxxx -- Display instruction operands
                                ;-------------------------------------------------------------------
                                ;  TYP1 -- Instruction has no operands.
                                ;
010026  000207                  ODC1:   RETURN
                                ;
                                ;-------------------------------------------------------------------
                                ;  TYP2 -- Operand has a single general operand.
                                ;
010030                          ODC2:
                                ;
                                ;  Tab over to operand field
                                ;
010030  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010034  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the operand
                                ;
010040  010200                          MOV     R2,R0           ;Get the operand code
010042  004737  011000'                 CALL    ODCGEN          ;Display general operand
                                ;
                                ;  Finished
                                ;
010046  000207                          RETURN
                                ;
                                ;-------------------------------------------------------------------
                                ;  TYP3 -- Single register operand.
                                ;
010050                          ODC3:
                                ;
                                ;  Tab over to the operand field
                                ;
010050  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010054  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the register
                                ;
010060  010200                          MOV     R2,R0           ;Get the register number
010062  004737  011472'                 CALL    ODCREG          ;Display the register number
010066  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP4 -- 4-bit absolute numeric operand
                                ;
010070                          ODC4:
                                ;
                                ;  Tab over to the operand field
                                ;
010070  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010074  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the value
                                ;
010100  010200                          MOV     R2,R0           ;Get value to display
010102  004737  006030'                 CALL    PRTOCT          ;Print the value
                                ;
                                ;  Finished
                                ;
010106  000207                          RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP5 -- Condition code modification instruction
                                ;
010110                          ODC5:
                                ;
                                ;  Determine which condition code is affected
                                ;
010110  032702  000001                  BIT     #1,R2           ;C-flag?
010114  001404                          BEQ     1$              ;Br if not
010116                                  .TTYOUT #'C
010126  032702  000002          1$:     BIT     #2,R2           ;V-flag?
010132  001404                          BEQ     2$              ;Br if not
010134                                  .TTYOUT #'V
010144  032702  000004          2$:     BIT     #4,R2           ;Z-flag?
010150  001404                          BEQ     3$              ;Br if not
010152                                  .TTYOUT #'Z
010162  032702  000010          3$:     BIT     #10,R2          ;N-flag?
010166  001404                          BEQ     4$              ;Br if not
010170                                  .TTYOUT #'N
                                ;
                                ;  Finished
                                ;
010200  005237  000000G         4$:     INC     D.PCOL          ;Advance column number
010204  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP6 -- Branch offset operand
                                ;
010206                          ODC6:
                                ;
                                ;  Tab over to the operand field
                                ;
010206  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010212  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the branch target
                                ;
010216  010200                          MOV     R2,R0           ;Get instruction word
010220  042700  177400                  BIC     #^C377,R0       ;Leave only branch offset
010224  032700  000200                  BIT     #200,R0         ;Is offset negative?
010230  001402                          BEQ     1$              ;Br if not
010232  052700  177400                  BIS     #177400,R0      ;Sign extend the offset
010236  006300                  1$:     ASL     R0              ;Convert word offset to byte offset
010240  060100                          ADD     R1,R0           ;Add current PC address
010242  004737  011720'                 CALL    ODCADR          ;Display the address
                                ;
                                ;  Finished
                                ;
010246  000207                          RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP7 -- Register and general destination.
                                ;
010250                          ODC7:
                                ;
                                ;  Tab over to the operand field
                                ;
010250  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010254  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the register
                                ;
010260  010200                          MOV     R2,R0           ;Get instruction word
010262  072027  177772                  ASH     #-6,R0          ;Right justify the register value
010266  004737  011472'                 CALL    ODCREG          ;Display the register name
010272                                  .TTYOUT #COMMA          ;Display ","
010302  005237  000006                  INC     D.PCOL          ;Inc column counter
                                ;
                                ;  Display the destination address
                                ;
010306  010200                          MOV     R2,R0           ;Get the destination address descriptor
010310  004737  011000'                 CALL    ODCGEN          ;Display the address
                                ;
                                ;  Finished
                                ;
010314  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP8 -- Register destination and general source operand (SS,R)
                                ;
010316                          ODC8:
                                ;
                                ;  Tab over to the operand field
                                ;
010316  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010322  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the source item
                                ;
010326  010200                          MOV     R2,R0           ;Get the source operand code
010330  004737  011000'                 CALL    ODCGEN          ;Display it
010334                                  .TTYOUT #COMMA          ;Display ","
010344  005237  000000G                 INC     D.PCOL          ;Advance column counter
                                ;
                                ;  Display the register destination
                                ;
010350  010200                          MOV     R2,R0           ;Get the register number
010352  072027  177772                  ASH     #-6,R0          ;Right justify
010356  004737  011472'                 CALL    ODCREG          ;Display the register
                                ;
                                ;  Finished
                                ;
010362  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP9 -- Two general operands
                                ;
010364                          ODC9:
                                ;
                                ;  Tab over to the operand field
                                ;
010364  012700  000010                MOV     #COLOPN,R0      ;Get operand column number
010370  004737  012064'               CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the source operand
                                ;
010374  010200                        MOV     R2,R0           ;Get instruction word
010376  072027  177772                ASH     #-6,R0          ;Right justify the source item
010402  004737  011000'               CALL    ODCGEN          ;Display source item
010406                                .TTYOUT #COMMA          ;Print a comma
                                ;
                                ;  Display the destination operand
                                ;
010416  010200                        MOV     R2,R0           ;Get destination item code
010420  004737  011000'               CALL    ODCGEN          ;Display destination item
                                ;
                                ;  Finished
                                ;
010424  000207                        RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP10 -- SOB instruction
                                ;
010426                          ODC10:
                                ;
                                ;  Tab over to the operand field
                                ;
010426  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010432  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the register number
                                ;
010436  010200                          MOV     R2,R0           ;Get instruction word
010440  072027  177772                  ASH     #-6,R0          ;Right justify register number
010444  004737  011472'                 CALL    ODCREG          ;Display the register name
010450                                  .TTYOUT #COMMA          ;Print comma
010460  005237  000000G                 INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the destination address
                                ;
010464  042702  177700                  BIC     #^C77,R2        ;Mask out all but branch offset
010470  010100                          MOV     R1,R0           ;Get the current PC address
010472  006302                          ASL     R2              ;Get 2* offset
010474  160200                          SUB     R2,R0           ;Get destination address
010476  004737  011720'                 CALL    ODCADR          ;Display the address
                                ;
                                ;  Finished
                                ;
010502  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP11 -- EMT and TRAP instructions
                                ;
010504                          ODC11:
                                ;
                                ;  Tab over to the operand field
                                ;
010504  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010510  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the argument value
                                ;
010514  010200                          MOV     R2,R0           ;Get instruction word
010516  042700  177400                  BIC     #^C377,R0       ;Mask out all but operand value
010522  004737  006030'                 CALL    PRTOCT          ;Print octal value
                                ;
                                ;  Finished
                                ;
010526  000207                          RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP12 -- Single floating-point operand
                                ;
010530                          ODC12:
                                ;
                                ;  Tab over to the operand field
                                ;
010530  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010534  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the floating point operand
                                ;
010540  010200                          MOV     R2,R0           ;Get operand code
010542  004737  011672'                 CALL    ODCFPU          ;Display general FPU operand
                                ;
                                ;  Finished
                                ;
010546  000207                          RETURN

                                ;-----------------------------------------------------------------------
                                ;  TYP13 -- Floating accumulator and floating source operand.
                                ;
010550                          ODC13:
                                ;
                                ;  Tab over to the operand field
                                ;
010550  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010554  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the source operand
                                ;
010560  010200                          MOV     R2,R0           ;Get instruction word
010562  004737  011672'                 CALL    ODCFPU          ;Display the source operand
010566                                  .TTYOUT #COMMA          ;Display comma
010576  005237  000000G                 INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the accumulator number
```

```
                                        ;
010602  010200                                MOV     R2,R0           ;Get accumulator number
010604  072027  177772                        ASH     #-6,R0          ;Right justify
010610  004737  011620'                       CALL    ODCACC          ;Display the accumulator number
                                        ;
                                        ;  Finished
                                        ;
010614  000207                                RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP14 -- Floating accumulator and general source operand.
                                ;
                                ODC14:
                                ;
                                ;  Tab over to the operand field
                                ;
 010616  012700  000010                 MOV     #COLOPN,R0      ;Get operand column number
 010622  004737  012064'                CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the source operand
                                ;
 010626  010200                         MOV     R2,R0           ;Get instruction word
 010630  004737  011000'                CALL    ODCGEN          ;Display the source operand
 010634                                 .TTYOUT #COMMA          ;Display comma
 010644  005237  000006                 INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the accumulator number
                                ;
 010650  010200                         MOV     R2,R0           ;Get accumulator number
 010652  072027  177772                 ASH     #-6,R0          ;Right justify
 010656  004737  011620'                CALL    ODCACC          ;Display the accumulator number
                                ;
                                ;  Finished
                                ;
 010662  000207                         RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP15 -- Floating accumulator and floating destination operand.
                                ;
010664                          ODC15:
                                ;
                                ;  Tab over to the operand field
                                ;
010664  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010670  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the floating accumulator
                                ;
010674  010200                          MOV     R2,R0           ;Get the accumulator number
010676  072027  177772                  ASH     #-6,R0          ;Right justify
010702  004737  011620'                 CALL    ODCACC          ;Display the accumulator
010706                                  .TTYOUT #COMMA          ;Display comma
010716  005237  000000G                 INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the floating destination
                                ;
010722  010200                          MOV     R2,R0           ;Get the destination operand code
010724  004737  011672'                 CALL    ODCFPU          ;Display it
                                ;
                                ;  Finished
                                ;
010730  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  TYP16 -- Floating accumulator and general destination operand.
                                ;
010732                          ODC16:
                                ;
                                ;  Tab over to the operand field
                                ;
010732  012700  000010                  MOV     #COLOPN,R0      ;Get operand column number
010736  004737  012064'                 CALL    ODCTAB          ;Tab over to operand field
                                ;
                                ;  Display the floating accumulator
                                ;
010742  010200                          MOV     R2,R0           ;Get the accumulator number
010744  072027  177772                  ASH     #-6,R0          ;Right justify
010750  004737  011620'                 CALL    ODCACC          ;Display the accumulator
010754                                  .TTYOUT #COMMA          ;Display comma
010764  005237  000000G                 INC     D.PCOL          ;Advance column number
                                ;
                                ;  Display the destination
                                ;
010770  010200                          MOV     R2,R0           ;Get the destination operand code
010772  004737  011000'                 CALL    ODCGEN          ;Display it
                                ;
                                ;  Finished
                                ;
010776  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.
                                ;
011000  010246                  ODCGEN: MOV     R2,-(SP)
011002  010346                          MOV     R3,-(SP)
                                ;
                                ;  Get the addressing mode
                                ;
011004  010002                          MOV     R0,R2
011006  072227  177776                  ASH     #-2,R2          ;Right justify 2* mode
011012  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.
                                ;
011016  010003                          MOV     R0,R3           ;Get mode and register value
011020  042703  177770                  BIC     #^C7,R3         ;Mask out all but register number
011024  020327  000007                  CMP     R3,#7           ;Is register R7?
011030  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.
                                ;
011032  004772  011452'                 CALL    @AD7VEC(R2)     ;Call the processing routine
011036  000402                          BR      9$
                                ;
                                ;  Jump to processing routine based on the mode
                                ;
011040  004772  011432'         1$:     CALL    @ADRVEC(R2)     ;Jump to processing routine
                                ;
                                ;  Finished
                                ;
011044  012603                  9$:     MOV     (SP)+,R3
011046  012602                          MOV     (SP)+,R2
011050  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Display routines for registers R0 through R6
                                ;
                                ;  Mode 0 -- R
                                ;
011052  004737  011472'         ADRMD0: CALL    ODCREG          ;Display register
011056  000207                          RETURN
                                ;
                                ;  Mode 1 -- (R)
                                ;
011060  010046                  ADRMD1: MOV     R0,-(SP)        ;Save the register number
011062                                  .TTYOUT #'(             ;Open paren
011072  012600                          MOV     (SP)+,R0        ;Recover the register number
011074  004737  011472'                 CALL    ODCREG          ;Display the register number
011100                                  .TTYOUT #')             ;Close paren
011110  000207                          RETURN
                                ;
                                ;  Mode 2 -- (R)+
                                ;
011112  004737  011060'         ADRMD2: CALL    ADRMD1          ;Print "(r)"
011116                                  .TTYOUT #'+             ;Put in trailing plus sign
011126  000207                          RETURN
                                ;
                                ;  Mode 3 -- @(R)+
                                ;
011130  010046                  ADRMD3: MOV     R0,-(SP)        ;Save the register number
011132                                  .TTYOUT #'@             ;Print "@"
011142  012600                          MOV     (SP)+,R0        ;Recover the register number
011144  004737  011112'                 CALL    ADRMD2          ;Print "(r)+"
011150  000207                          RETURN
                                ;
                                ;  Mode 4 --  -(R)
                                ;
011152  010046                  ADRMD4: MOV     R0,-(SP)        ;Save the register number
011154                                  .TTYOUT #'-             ;Print "-"
011164  012600                          MOV     (SP)+,R0        ;Recover the register number
011166  004737  011060'                 CALL    ADRMD1          ;Print "(r)"
011172  000207                          RETURN
                                ;
                                ;  Mode 5 -- @-(R)
                                ;
011174  010046                  ADRMD5: MOV     R0,-(SP)        ;Save the register number
011176                                  .TTYOUT #'@             ;Print "@-"
011206                                  .TTYOUT #'-
011216  012600                          MOV     (SP)+,R0        ;Recover register number
011220  004737  011060'                 CALL    ADRMD1          ;Print "(r)"
011224  000207                          RETURN
                                ;
                                ;  Mode 6 -- X(R)
                                ;
011226  010046                  ADRMD6: MOV     R0,-(SP)        ;Save the register number
011230  106521                          MFPD    (R1)+           ;Get the vector base address
011232  062737  000002  000000G         ADD     #2,.ILEN        ;Say instruction uses 2 more bytes
011240  012600                          MOV     (SP)+,R0        ;Get value to R0
011242  004737  011720'                 CALL    ODCADR          ;Display the address
011246  012600                          MOV     (SP)+,R0        ;Recover the register number
011250  004737  011060'                 CALL    ADRMD1          ;Print "(r)"
```

```
     58 011254  000207                          RETURN
     59                                ;
     60                                ;  Mode 7 -- @X(R)
     61                                ;
     62 011256  010046                 ADRMD7:  MOV      R0,-(SP)        ;Save register number
     63 011260                                  .TTYOUT #'@               ;Print "@"
     64 011270  012600                          MOV      (SP)+,R0        ;Recover register number
     65 011272  004737  011226'                 CALL     ADRMD6          ;Print "x(r)"
     66 011276  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Display routines for the various modes with R7.
                                ;
                                ;  Mode 2 -- #n
                                ;
011300                          AD7MD2: .TTYOUT #'#             ;Print "#"
011310  106521                          MFPD    (R1)+           ;Get the value onto the stack
011312  062737  000002  000000G       ADD     #2,D.ILEN       ;Say instruction uses 2 more bytes
011320  012600                        MOV     (SP)+,R0        ;Get the value
011322  004737  012022'               CALL    ODCNEG          ;See if value should be printed as neg number
011326  103002                        BCC     9$              ;Br if it was a negative number
011330  004737  006030'               CALL    PRTOCT          ;Print as an octal value
011334  000207                  9$:   RETURN
                                ;
                                ;  Mode 3 -- @#address
                                ;
011336                          AD7MD3: .TTYOUT #'@             ;Print "@#"
011346                                  .TTYOUT #'#
011356  106521                        MFPD    (R1)+           ;Get the address
011360  062737  000002  000000G       ADD     #2,D.ILEN       ;Advance the instruction length
011366  012600                        MOV     (SP)+,R0        ;Get the address
011370  004737  011720'               CALL    ODCADR          ;Display the address
011374  000207                        RETURN
                                ;
                                ;  Mode 6 -- address
                                ;
011376  106521                  AD7MD6: MFPD    (R1)+           ;Get the address
011400  062737  000002  000000G       ADD     #2,D.ILEN       ;Advance the instruction length
011406  012600                        MOV     (SP)+,R0        ;Get the address
011410  060100                        ADD     R1,R0           ;Relocate the address
011412  004737  011720'               CALL    ODCADR          ;Display the address
011416  000207                        RETURN
                                ;
                                ;  Mode 7 -- @address
                                ;
011420                          AD7MD7: .TTYOUT #'@             ;Print "@"
011430  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 011432  011052'                 ADRVEC: .WORD   ADRMD0           ;Mode 0
      6 011434  011060'                         .WORD   ADRMD1           ;Mode 1
      7 011436  011112'                         .WORD   ADRMD2           ;Mode 2
      8 011440  011130'                         .WORD   ADRMD3           ;Mode 3
      9 011442  011152'                         .WORD   ADRMD4           ;Mode 4
     10 011444  011174'                         .WORD   ADRMD5           ;Mode 5
     11 011446  011226'                         .WORD   ADRMD6           ;Mode 6
     12 011450  011256'                         .WORD   ADRMD7           ;Mode 7
     13                                 ;
     14                                 ;  Jump table used to dispatch to the correct display routine for
     15                                 ;  addressing modes using register R7 (PC).
     16                                 ;
     17 011452  011052'                 AD7VEC: .WORD   ADRMD0           ;Mode 0
     18 011454  011060'                         .WORD   ADRMD1           ;Mode 1
     19 011456  011300'                         .WORD   AD7MD2           ;Mode 2
     20 011460  011336'                         .WORD   AD7MD3           ;Mode 3
     21 011462  011152'                         .WORD   ADRMD4           ;Mode 4
     22 011464  011174'                         .WORD   ADRMD5           ;Mode 5
     23 011466  011376'                         .WORD   AD7MD6           ;Mode 6
     24 011470  011420'                         .WORD   AD7MD7           ;Mode 7
```

```
                                        .SBTTL  ODCREG -- Display register name
                                ;-----------------------------------------------------------------------
                                ;  ODCREG is called to display the name of a register.
                                ;
                                ;  Inputs:
                                ;   R0 = Register number
                                ;
011472  010046                  ODCREG: MOV     R0,-(SP)
                                ;
                                ;  Determine which register this is
                                ;
011474  042700  177770                  BIC     #^C7,R0         ;Clear all but the register number
                                ;
                                ;  See if this is R7 (PC)
                                ;
011500  020027  000007                  CMP     R0,#7           ;Is register PC?
011504  001011                          BNE     1$
011506                                  .TTYOUT #'P             ;Display "PC"
011516                                  .TTYOUT #'C
011526  000427                          BR      9$
                                ;
                                ;  See if this is R6 (SP)
                                ;
011530  020027  000006          1$:     CMP     R0,#6           ;Is register SP?
011534  001011                          BNE     2$              ;Br if not
011536                                  .TTYOUT #'S             ;Print "SP"
011546                                  .TTYOUT #'P
011556  000413                          BR      9$
                                ;
                                ;  This is a register in the range R0 to R5.
                                ;
011560                          2$:     .TTYOUT #'R             ;Print "Rn"
011570  011600                          MOV     (SP),R0         ;Get register number
011572  042700  177770                  BIC     #^C7,R0
011576  062700  000060                  ADD     #'0,R0          ;Convert to ascii digit
011602                                  .TTYOUT                 ;Print it
                                ;
                                ;  Finished
                                ;
011606  062737  000002  000000G 9$:     ADD     #2,@#.PCOL      ;Advance print column counter
011614  012600                          MOV     (SP)+,R0
011616  000207                          RETURN
```

```
                                        .SBTTL  ODCACC -- Print a floating point accumulator name
                                ;-----------------------------------------------------------------------
                                ;  Print the name of a floating point accumulator.
                                ;
                                ;  Inputs:
                                ;   R0 = Accumulator number
                                ;
011620  010046                  ODCACC: MOV     R0,-(SP)
                                ;
                                ;  Print ACn
                                ;
011622                                  .TTYOUT #'A
011632                                  .TTYOUT #'C
011642  011600                          MOV     (SP),R0
011644  042700  177774                  BIC     #^C3,R0         ;Clear all but ACC #
011650  062700  000060                  ADD     #'0,R0          ;Convert to ascii char
011654                                  .TTYOUT                 ;Print the #
                                ;
                                ;  Finished
                                ;
011660  062737  000003  000000G         ADD     #3,D.PCOL       ;Advance print column number
011666  012600                          MOV     (SP)+,R0
011670  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 #.
                                ;
011672  010046                  ODCFPU: MOV     R0,-(SP)
                                ;
                                ;  Determine if it is a floating point accumulator
                                ;
011674  032700  000070                  BIT     #70,R0          ;Is mode = 0 (accumulator)?
011700  001003                          BNE     1$              ;Br if not
011702  004737  011620'                  CALL    ODCACC          ;Display an accumulator name
011706  000402                          BR      9$
                                ;
                                ;  It is not an accumulator
                                ;
011710  004737  011000'          1$:     CALL    ODCGEN          ;Display general operand
                                ;
                                ;  Finished
                                ;
011714  012600                  9$:     MOV     (SP)+,R0
011716  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.
                                ;
011720  010046                  ODCADR: MOV     R0,-(SP)
011722  010346                          MOV     R3,-(SP)
                                ;
                                ;  Determine if address should be shown as a negative number
                                ;
011724  004737  012022'                 CALL    ODCNEG          ;Should this be shown as a negative number?
011730  103031                          BCC     9$              ;Br if yes
                                ;
                                ;  Determine if address is relative to a relocation base
                                ;
011732  032737  000000G 000000G        BIT     #DP$DAA,D.PFMT  ;Should address be absolute?
011740  001023                          BNE     1$              ;Br if yes
011742  004737  007412'                 CALL    RELADR          ;See if address is within relocation region
011746  103420                          BCS     1$              ;Br if address is not within relocation region
                                ;
                                ;  Address is within a relocation region.
                                ;  Display in the form "[r,address]".
                                ;
011750  010003                          MOV     R0,R3           ;Save the address
011752                                  .TTYOUT #'[             ;Print "[r,address]"
011762  010300                          MOV     R3,R0           ;Get back address
011764  004737  007322'                 CALL    SHOADR          ;Display the address
011770                                  .TTYOUT #']             ;Put in closing bracket
012000  062737  000002  000000G        ADD     #2,D.PCOL       ;Advance print column counter
012006  000402                          BR      9$
                                ;
                                ;  Address is not relative to a relocation base
                                ;
012010  004737  006030'         1$:     CALL    PRTOCT          ;Display the address
                                ;
                                ;  Finished
                                ;
012014  012603                  9$:     MOV     (SP)+,R3
012016  012600                          MOV     (SP)+,R0
012020  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.
                                ;
012022                          ODCNEG:
                                ;
                                ;  See if this is a value to be displayed as a negative number
                                ;
012022  020027  177776                  CMP     R0,#177776      ;Is this a negative value?
012026  103414                  BLO     9$              ;Br if not
                                ;
                                ;  This is a negative value
                                ;
012030  010046                  MOV     R0,-(SP)
012032                          .TTYOUT #'-             ;Print minus sign
012042  011600                  MOV     (SP),R0         ;Get the value
012044  005400                  NEG     R0              ;Negate it
012046  004737  006030'          CALL    PRTOCT          ;Print it
012052  012600                  MOV     (SP)+,R0        ;Recover original value
012054  000241                  CLC                     ;This is a negative number
012056  000401                  BR      10$
                                ;
                                ;  Finished
                                ;
012060  000261                  9$:     SEC             ;This is not a negative value
012062  000207                  10$:    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.
                                ;
012064  010046                  ODCTAB: MOV     R0,-(SP)
                                ;
                                ;  Print spaces until we reach the desired column
                                ;
012066                          1$:     .TTYOUT #SPACE          ;Print a space
012076  005237  000000G                 INC     D.PCOL          ;Advance column number
012102  023716  000000G                 CMP     D.PCOL,(SP)     ;Reached desired column?
012106  103767                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
012110  012600                          MOV     (SP)+,R0
012112  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
                                ;
012114                          INSBAS:
                                ;        Mask    Code    Type    Name
                                ;       ------  ------  ----    ----
012114                          INSTR   000000, 000000, TYP1,   HALT
012126                          INSTR   000000, 000001, TYP1,   WAIT
012140                          INSTR   000000, 000002, TYP1,   RTI
012152                          INSTR   000000, 000003, TYP1,   BPT
012164                          INSTR   000000, 000004, TYP1,   IOT
012176                          INSTR   000000, 000005, TYP1,   RESET
012212                          INSTR   000000, 000006, TYP1,   RTT
```

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

```
    115 013312                          INSTR   000377, 104000, TYP11, EMT
    116 013324                          INSTR   000377, 104400, TYP11, TRAP
    117 013336                          INSTR   000077, 105000, TYP2,  CLRB
    118 013350                          INSTR   000077, 105100, TYP2,  COMB
    119 013362                          INSTR   000077, 105200, TYP2,  INCB
    120 013374                          INSTR   000077, 105300, TYP2,  DECB
    121 013406                          INSTR   000077, 105400, TYP2,  NEGB
    122 013420                          INSTR   000077, 105500, TYP2,  ADCB
    123 013432                          INSTR   000077, 105600, TYP2,  SBCB
    124 013444                          INSTR   000077, 105700, TYP2,  TSTB
    125 013456                          INSTR   000077, 106000, TYP2,  RORB
    126 013470                          INSTR   000077, 106100, TYP2,  ROLB
    127 013502                          INSTR   000077, 106200, TYP2,  ASRB
    128 013514                          INSTR   000077, 106300, TYP2,  ASLB
    129 013526                          INSTR   000077, 106500, TYP2,  MFPD
    130 013540                          INSTR   000077, 106600, TYP2,  MTPD
    131 013552                          INSTR   007777, 110000, TYP9,  MOVB
    132 013564                          INSTR   007777, 120000, TYP9,  CMPB
    133 013576                          INSTR   007777, 130000, TYP9,  BITB
    134 013610                          INSTR   007777, 140000, TYP9,  BICB
    135 013622                          INSTR   007777, 150000, TYP9,  BISB
    136 013634                          INSTR   007777, 160000, TYP9,  SUB
    137 013646                          INSTR   000000, 170000, TYP1,  CFCC
    138 013660                          INSTR   000000, 170001, TYP1,  SETF
    139 013672                          INSTR   000000, 170002, TYP1,  SETI
    140 013704                          INSTR   000000, 170011, TYP1,  SETD
    141 013716                          INSTR   000000, 170012, TYP1,  SETL
    142 013730                          INSTR   000077, 170100, TYP2,  LDFPS
    143 013744                          INSTR   000077, 170200, TYP2,  STFPS
    144 013760                          INSTR   000077, 170300, TYP2,  STST
    145 013772                          INSTR   000077, 170400, TYP12, CLRF
    146 014004                          INSTR   000077, 170500, TYP12, TSTF
    147 014016                          INSTR   000077, 170600, TYP12, ABSF
    148 014030                          INSTR   000077, 170700, TYP12, NEGF
    149 014042                          INSTR   000377, 171000, TYP13, MULF
    150 014054                          INSTR   000377, 171400, TYP13, MODF
    151 014066                          INSTR   000377, 172000, TYP13, ADDF
    152 014100                          INSTR   000377, 172400, TYP13, LDF
    153 014112                          INSTR   000377, 173000, TYP13, SUBF
    154 014124                          INSTR   000377, 173400, TYP13, CMPF
    155 014136                          INSTR   000377, 174000, TYP15, STF
    156 014150                          INSTR   000377, 174400, TYP13, DIVF
    157 014162                          INSTR   000377, 175000, TYP16, STEXP
    158 014176                          INSTR   000377, 175400, TYP16, STCFI
    159 014212                          INSTR   000377, 176000, TYP16, STCFD
    160 014226                          INSTR   000377, 176400, TYP14, LDEXP
    161 014242                          INSTR   000377, 177000, TYP14, LDCIF
    162 014256                          INSTR   000377, 177400, TYP13, LDCDF
    163 014272                          INSTR   000077, 106400, TYP2,  MTPS
    164 014304                          INSTR   000077, 106700, TYP2,  MFPS
    165 014316                          INSTR   177777, 000000, TYP1,  ????
    166 014330                          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
                                ;
 014330                         TYPVEC:
 014330                                 TYPDEF  TYP1,ODC1       ;No operand
 014332                                 TYPDEF  TYP2,ODC2       ;Single general operand.
 014334                                 TYPDEF  TYP3,ODC3       ;Single register operand (RTS)
 014336                                 TYPDEF  TYP4,ODC4       ;4-bit absolute numeric operand.
 014340                                 TYPDEF  TYP5,ODC5       ;Condition code modification instruction.
 014342                                 TYPDEF  TYP6,ODC6       ;Branch offset operand.
 014344                                 TYPDEF  TYP7,ODC7       ;Register source, general operand destination.
 014346                                 TYPDEF  TYP8,ODC8       ;Register destination, general source operand
 014350                                 TYPDEF  TYP9,ODC9       ;General source and destination operands.
 014352                                 TYPDEF  TYP10,ODC10     ;Register and loop offset (SOB).
 014354                                 TYPDEF  TYP11,ODC11     ;EMT and TRAP instructions.
 014356                                 TYPDEF  TYP12,ODC12     ;One floating point operand.
 014360                                 TYPDEF  TYP13,ODC13     ;Floating accum and floating source operand.
 014362                                 TYPDEF  TYP14,ODC14     ;Floating accum and general source operand.
 014364                                 TYPDEF  TYP15,ODC15     ;Floating accumulator and floating destination
 014366                                 TYPDEF  TYP16,ODC16     ;Floating accumulator and general destination.

        000001                          .END
Errors detected:  0
*** Assembler statistics

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

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

Cross reference table (CREF V05.05)

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

```
        53-8      54-8      55-8
COMMA    1-76#    48-41     49-15     50-16     51-16     52-54     53-15     54-16     55-16
CORUSR   1-32     6-15      43-15
CR       1-70#    1-92      1-94      2-19      18-15     18-25     18-40     20-27     43-29
CRCHAR   1-94#    20-32     21-19
CRLF     1-93#    5-31      5-39      9-5       12-9      16-8      16-19     17-65     21-14     22-22     22-27     27-7
        27-28     27-41     27-56     27-68     35-6
D$BKST   1-30     24-8      24-26     25-21
D$CKBK   1-33     17-60     24-10     25-7      26-53     28-19     28-24
D$DBRK   1-28     5-43      7-14      26-26     26-49     27-11
D$DMON   1-29     7-22      7-65      8-6       8-12      8-28      8-54      19-5      19-8      26-30
D$DVAL   1-30     19-5      19-17     26-37
D$FBRK   1-31     6-28      7-14      7-77      8-5       27-54
D$IBRK   1-28     5-43      7-14      7-68      26-26     26-85     27-39
D$INIT   1-34     6-22      23-16
D$IPND   1-30     7-21      7-64      7-71      8-8       8-12      8-54      26-59
D$ISPC   1-36     16-39     16-44     37-33     37-45     37-58     38-28     38-40
D$RUN    1-33     7-39      8-10
D$SBRK   1-28     5-43      7-14      26-26     26-66     27-32
D$SSTP   1-28     7-22      7-65      8-6       8-12      8-54      16-26     16-31     17-64     26-53     26-64
D$TSTP   1-34     7-22      7-65      7-84      8-6       8-12      8-54      17-30     26-53     26-64
D. BKAD  1-25     3-33      15-13     15-21*    24-16     25-13     26-75     28-12
D. BKNM  1-28     26-67*    26-84*    27-42     27-47
D. BKSV  1-29     24-21     25-16*
D. BYTM  1-25     10-9*     10-22*    20-24     21-11     22-19     37-26     37-67     38-22     38-54
D. CBRK  1-35     17-59*    27-44*
D. DADR  1-29     8-30      19-10*    26-32     27-17     27-20
D. DOLD  1-29     8-35*     26-40     27-23*    27-27*
D. DTRG  1-29     19-18*    26-44
D. END   1-31     23-13
D. FLAG  1-28     5-43*     6-22      6-28*     7-14      7-21*     7-22      7-39*     7-64*     7-65      7-68      7-71*
         7-77     7-84*     8-5*      8-6       8-8*      8-10*     8-12      8-28      8-54      16-26*    16-31*    16-39*
        16-44*    17-30*    17-60*    17-64*    19-5*     19-8*     19-17*    23-16*    24-8      24-10     24-26*    25-7
        25-21*    26-26*    26-30     26-37     26-49*    26-53     26-59     26-64     26-66*    26-85*    27-11     27-32
        27-39     27-54     28-19*    28-24*    37-33     37-45     37-58     38-28     38-40
D. ILEN  1-34     10-10*    10-23*    18-9*     21-13     45-20*    57-53*    58-8*     58-20*    58-28*
D. LOC   1-25     5-54*     8-9*      10-7      11-12     12-7*     12-33     13-22     13-29     20-10*    20-18     20-26*
        20-31*    21-9      21-13*    21-18*    22-12     22-21*    22-26*    36-16*    37-32     37-43     37-50     37-72*
        37-78     37-80     38-27     38-38     38-46     38-59     38-64     38-67     39-23     39-29
D. LOCM  1-26     11-18     12-8*     36-15*    37-16     37-21     38-17     39-18
D. LVAL  1-33     10-12*    10-25*    12-6      12-32     14-36     14-51     18-6
D. MASK  1-33     3-31      8-32      23-15*    26-34     27-25
D. NMBE  1-31     30-38
D. NMBF  1-25     30-28     30-54
D. PCNT  1-32     7-75      7-79*     16-9*     16-18*
D. PCOL  1-34     10-33*    11-28     18-53*    20-5*     20-17*    21-5*     22-11*    31-19*    32-23*    32-28*    33-18*
        33-43*    34-27*    35-8*     35-23*    39-50*    39-53*    39-60*    41-28*    43-24*    43-33*    45-25*    45-51*
        47-42*    48-42*    49-16*    51-17*    52-55*    53-16*    54-17*    55-17*    60-40*    61-21*    63-35*    65-17*
        65-18
D. PFMT  1-35     3-32      40-16     63-22
D. PS    1-30     3-30      4-17*     5-26*     7-35*     8-53*     8-56*     8-57*     8-61
D. 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
D.R7      1-27    3-29    4-16*   5-25*   5-37    7-8*    7-34*   7-70*   8-62    16-7*   17-26*  17-37
         26-73   27-14   27-61   27-66   29-62
D.RLBS    1-25    3-34   13-23   14-23*  14-37   29-55   41-23   42-23   42-25   42-28
D.SPSV    1-31    5-52    7-51*  35-9
D.STAR    1-31   23-11
D.SVCH    1-25   31-12   31-31*  31-41*
D.V1FL    1-26    9-13*   9-22*  11-9    13-11   14-29   16-5    16-16   17-13   17-15*  18-57*  20-6
         20-20   21-6    22-15   36-13
D.V2FL    1-26    9-15*   9-33*  14-13   15-5    16-27   16-40   19-15
D.VAL1    1-26    9-14*   9-21*  13-10   14-23   14-31   15-19   16-7    16-18   17-21   19-6    19-9
         20-8    20-22   22-17   36-16
D.VAL2    1-26    9-16*   9-32*  14-18   15-7    16-29   16-42   19-18
DBGBRK    1-21    6-10#
DBGENT    1-21    4-8#
DBGINI    4-12    6-24   23-7#
DBGOFF    8-11   44-13#
DBGON     7-88   44-5#
DBGRUN    4-21    5-46    7-39#
DBGTRP    1-21    5-13#
DECODE   11-24   27-67   45-12#
DOPRT     5-31    5-39    9-5     12-9    16-8    16-19   17-65   20-32   21-14   21-19   22-22   22-27
         23-25   27-7    27-28   27-41   27-56   27-68   35-6    35-7    43-9#
DP$DAA    1-35   63-22
DP$LAA    1-35   40-16
EM$IIV    1-89#  29-74
EM$IRB    1-90#  29-51
EM$IRC    1-86#  18-34
EM$IVC    1-87#   9-43
EM$IVL    1-88#  14-21   14-34   15-11
EM$NTL    1-91#  30-80
ERRPRT    9-43   14-21   14-34   15-11   18-34   29-51   29-74   30-80   35-6#
EXIT1     7-80    8-5#   16-10   16-20   17-66
EXIT2     8-49#
FLGBRK   15-25   28-7#
GETCHR    9-26    9-37   18-13   18-24   18-39   18-54   29-19   29-53   29-68   30-17   30-30   30-71
         31-8#
GETCMD    9-13#  10-15   10-35   11-35   13-35   14-57
GETVAL   10-11   10-24   38-13#
IB$LEN   45-39   45-50   66-17#
IB$MSK   45-36   45-61   66-14#
IB$NAM   45-42   45-53   66-18#
IB$TYP   45-70   66-16#
IB$VAL   45-37   66-15#
INSBAS   45-34   66-48#
INSEND  66-166#
INSLEN   17-57   17-70#
INTADR    3-22#  29-76   39-30   39-47
INTCHR    3-4#    3-17   29-70   39-51
INTEND    3-35#  39-42
LF        1-71#   1-92    2-20   18-17   18-27   18-42   18-55   43-31
LSTADR    5-38   27-15   27-18   27-62   39-24   40-12#
LSTTXT    5-34    5-36    9-9    10-14   10-28   10-34   12-19   12-25   13-17   13-20   13-31   14-28
         14-42   14-45   14-53   18-5    18-8    20-29   20-34   21-16   21-21   22-24   22-29   27-13
         27-16   27-19   27-34   27-46   27-49   27-57   35-20#  35-24   45-24
```

```
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     16-45     18-62     19-22     20-36     21-23     22-14
NUMCMD    2-24#     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    1-66#    44-13
ONEMT     1-65#    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#
```

Cross reference table (CREF V05.05)

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

Cross reference table (CREF V05.05)

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