```
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  LLL              TTTTTTTTTTTTTTTT    AAAAAAAA
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  LLL              TTTTTTTTTTTTTTTT    AAAAAAAA
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  LLL              TTTTTTTTTTTTTTTT    AAAAAAAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDD         DDD  EEEEEEEEEEEE     LLL                      TTT      AAA         AAA
DDD         DDD  EEEEEEEEEEEE     LLL                      TTT      AAA         AAA
DDD         DDD  EEEEEEEEEEEE     LLL                      TTT      AAA         AAA
DDD         DDD  EEE              LLL                      TTT      AAAAAAAAAAAAAAA
DDD         DDD  EEE              LLL                      TTT      AAAAAAAAAAAAAAA
DDD         DDD  EEE              LLL                      TTT      AAAAAAAAAAAAAAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDD         DDD  EEE              LLL                      TTT      AAA         AAA
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  LLLLLLLLLLLLLLL          TTT      AAA         AAA
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  LLLLLLLLLLLLLLL          TTT      AAA         AAA
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  LLLLLLLLLLLLLLL          TTT      AAA         AAA
```

FILEID**XDELTA   N 3

```
XX      XX DDDDDDDD   EEEEEEEEEE LL         TTTTTTTTTT   AAAAAA
XX      XX DDDDDDDD   EEEEEEEEEE LL         TTTTTTTTTT   AAAAAA
XX      XX DD      DD EE         LL             TT     AA      AA
XX      XX DD      DD EE         LL             TT     AA      AA
  XX  XX   DD      DD EE         LL             TT     AA      AA
  XX  XX   DD      DD EE         LL             TT     AA      AA
    XX     DD      DD EEEEEEEE   LL             TT     AA      AA
    XX     DD      DD EEEEEEEE   LL             TT     AA      AA
  XX  XX   DD      DD EE         LL             TT     AAAAAAAAAA
  XX  XX   DD      DD EE         LL             TT     AAAAAAAAAA
XX      XX DD      DD EE         LL             TT     AA      AA    ....
XX      XX DD      DD EE         LL             TT     AA      AA    ....
XX      XX DDDDDDDD   EEEEEEEEEE LLLLLLLLLL     TT     AA      AA    ....
XX      XX DDDDDDDD   EEEEEEEEEE LLLLLLLLLL     TT     AA      AA    ....


LL          IIIIII     SSSSSSSS
LL          IIIIII     SSSSSSSS
LL            II     SS
LL            II     SS
LL            II     SS
LL            II     SS
LL            II       SSSSSS
LL            II       SSSSSS
LL            II             SS
LL            II             SS
LL            II             SS
LL            II             SS
LLLLLLLLLL  IIIIII   SSSSSSSS
LLLLLLLLLL  IIIIII   SSSSSSSS
```

Table of contents

```
0000          .IF     DF,SW_PROCESS                                   ;
0000          .TITLE  DELTA - MULTIMODE PROCESS DEBUGGER
0000          .IFF                                                    ;
0000          .TITLE  XDELTA - EXECUTIVE DEBUGGER
0000          .ENDC                                                   ;
0000          .IDENT  'V04-000'
0000
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000
0000   ;++
0000   ; FACILITY: EXECUTIVE, DEBUGGING TOOLS
0000   ;
0000   ; ABSTRACT:
0000   ;       THIS MODULE PRODUCES TWO DIFFERENT DEBUGGERS DEPENDING ON THE SETTING
0000   ;       OF THE ASSEMBLY SWITCH, SW_PROCESS.  DELTA IS A MULTIMODE PROCESS
0000   ;       DEBUGGER USING SYSTEM SERVICES WHILE XDELTA IS A STANDALONE EXEC
0000   ;       DEBUGGING TOOL.
0000   ;
0000   ;       COMMAND SYNTAX IS IDENTICAL FOR BOTH VERSIONS EXCEPT FOR ENVIRONMENTAL
0000   ;       DIFFERENCES.  THE SYNTAX IS QUITE TERSE AND SOMEWHAT CRYPTIC AND
0000   ;       IS DOCUMENTED IN THE ''GUIDE TO WRITING AN I/O DRIVER''.
0000   ;
0000   ; ENVIRONMENT:
0000   ;       DELTA - NORMAL PROCESS ENVIRONMENT, VARIOUS ACCESS MODES.
0000   ;       XDELTA - STANDALONE, RESIDENT, KERNEL MODE, IPL=31
0000   ;       BOTH VERSIONS MUST BE POSITION INDEPENDENT - BEWARE!
0000   ;--
```

```
0000            .SBTTL  HISTORY                 ; DETAILED
0000    ;
0000    ; AUTHOR:       R. HUSTVEDT       CREATION DATE: 15-NOV-76
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-016 WHM0001           Bill Matthews           18-Jul-1984
0000    ;               Call CON$GETCHAR and CON$PUTCHAR to do I/O to the console
0000    ;               terminal. Call CON$OWNCTY to allocate and CON$REALEASECTY to
0000    ;               release the console terminal.
0000    ;
0000    ;       V03-015 MSH0039           Michael S. Harvey       1-May-1984
0000    ;               Adjust image activation SET exception vector index
0000    ;               when setting up a DELTA rundown vector so that it
0000    ;               won't be lost by a subsequent image activation prior
0000    ;               to actual rundown.
0000    ;
0000    ;       V03-014 MSH0002           Michael S. Harvey       16-Jan-1984
0000    ;               Reenable AST delivery in EXIT command to ensure process
0000    ;               doesn't hang up when EXIT issued from kernel mode. Also,
0000    ;               lengthen input command buffer to match specified maximum
0000    ;               length in input QIO.
0000    ;
0000    ;       V03-013 TCM0003           Trudy C. Matthews       13-Dec-1983
0000    ;               Use "Write enable bit" when enabling and disabling
0000    ;               console terminal access for venus.
0000    ;
0000    ;       V03-012 KDM0084           Kathleen D. Morse       27-Sep-1983
0000    ;               Add MicroVAX I support to CPUDISP macros.
0000    ;
0000    ;       V03-011 RLRCPUDISP        Robert L. Rappaport     15-Jun-1983
0000    ;               Recode CPUDISP macros to use new format.
0000    ;
0000    ;       V03-010 MIR1039           Michael I. Rosenblum    27-May-1983
0000    ;               Fix non PIC reference in New format QIO
0000    ;
0000    ;       V03-009 MIR0039           Michael I. Rosenblum    29-Apr-1983
0000    ;               Make the process based DELTA use itemlist qio's with
0000    ;               The no editing bit set.
0000    ;
0000    ;       V03-008 JLV0236           Jake VanNoy             25-MAR-1983
0000    ;               Make QIO a QIOW in OUTZSTRING so that a read will
0000    ;               not block write.
0000    ;
0000    ;       V03-007 TCM0002           Trudy C. Matthews       16-Feb-1983
0000    ;               Correct console enable mask in TCM0001.
0000    ;
0000    ;       V03-006 ROW0159           Ralph O. Weber          28-JAN-1983
0000    ;               Enhance DELTA initialization to set all pages in DELTA to user
0000    ;               writable.  This corrects a problem encountered while trying to
0000    ;               debug DCL with DELTA.  It also guarantees that DELTA will work
0000    ;               in all access modes.  Change limit on rundown handler vector
0000    ;               table from 505 to <256-7>.
0000    ;
0000    ;       V03-005 TCM0001           Trudy C. Matthews       11-Jan-1983
0000    ;               Change 11/780 machine check handler to write PR$_SBIFS back
0000    ;               to itself to clear error bit.  Add 11/790 machine check
```

```
0000  ;               handler; initialize 11/790 console interface registers.
0000  ;
0000  ;       V03-004 ROW0143         Ralph O. Weber          24-NOV-1982
0000  ;               Change process-mode OUTZSTRING to do single QIO for whole
0000  ;               string.  Make terminal read/write QIOs do a $WAITEF and retry
0000  ;               if insufficient resources error is returned by the QIO system
0000  ;               service.  Make reference to CTL$GL_USRUNDWN in SETRUNDWN a
0000  ;               weak reference so that DELTA can be linked with SYSINIT.
0000  ;               Fix numerous branch destinations broken by the above.  Add
0000  ;               call to $IODEF definition macro.
0000  ;
0000  ;       V03-003 ACG0290         Andrew C. Goldstein,    5-May-1982  20:01
0000  ;               Condition rundown handler on user mode startup
0000  ;
0000  ;       V03-002 ACG0286         Andrew C. Goldstein,    13-Apr-1982  15:12
0000  ;               Use privileged rundown handler to reset exception vectors
0000  ;
0000  ;       V03-001 RIH0097         Richard I. Hustvedt     1-Apr-1982
0000  ;               Turn off processor register mode when proceeding.
0000  ;
0000
```

```
                  0000          .SBTTL  DECLARATIONS
                  0000
                  0000  ;
                  0000  ; INCLUDE FILES:
                  0000  ;
                  0000          $ACBDEF                         ; DEFINE AST CONTROL BLOCK
                  0000          $CADEF                          ; DEFINE ASSEMBLY SWITCHES
                  0000          $CLIDEF                         ; DEFINE CLI VALUES
                  0000          $IODEF                          ; DEFINE I/O FUNCTION CODES
                  0000          $IPLDEF                         ; DEFINE IPL VALUES
                  0000          $IRPDEF                         ; DEFINE IRP VALUES
                  0000          $PCBDEF                         ; DEFINE PROCESS CONTROL BLOCK
                  0000          $PRDEF                          ; DEFINE PROCESSOR REGISTERS
                  0000          $PRIDEF                         ; DEFINE PRIORITY INCREMENT CLASSES
                  0000          $PRTDEF                         ; DEFINE PROTECTION VALUES
                  0000          $PSLDEF                         ; DEFINE PSL FIELDS
                  0000          $SSDEF                          ; DEFINE SYSTEM SERVICE STATUS CODES
                  0000          $TRMDEF                         ; TERMINAL ITEMLIST DEFINITIONS
                  0000
                  0000  ;
                  0000  ; MACROS:
                  0000  ;
                  0000
                  0000  ;
                  0000  ; EQUATED SYMBOLS:
                  0000  ;
        00000008  0000  V_F1=8                                  ; FIELD 1 PRESENT FLAG
        00000009  0000  V_F2=9                                  ; FIELD 2 PRESENT FLAG
        0000000A  0000  V_F3=10                                 ; FIELD 3 PRESENT FLAG
        0000000B  0000  V_F4=11                                 ; FIELD 4 PRESENT FLAG
        0000000C  0000  V_F5=12                                 ; FIELD 5 PRESENT FLAG
        0000000D  0000  V_INSTR=13                              ; INSTRUCTION DISPLAY MODE
                  0000                                          ; (OVERRIDES HEX OR ASCII & CURTYPE)
                  0000
        00000000  0000  V_OPEN=0                                ; OPEN CELL FLAG
        00000001  0000  V_ASCII=1                               ; ASCII
        00000002  0000  V_INFIELD=2                             ; FIELD IN PROGRESS
        00000003  0000  V_TBIT=3                                ; ENABLE TBIT
        00000004  0000  V_ATBRK=4                               ; AT BREAKPOINT
        00000005  0000  V_TBITOK=5                              ; TBIT EXPECTED
        00000006  0000  V_RUB=6                                 ; RUBOUT IN PROGRESS
        00000007  0000  V_NEGATE=7                              ; NEGATE BIT
        0000000F  0000  V_PRMODE=15                             ; PROCESSOR REGISTER MODE
        0000001F  0000  V_PREG=31                               ; PROCESSOR REGISTER FLAG
                  0000
        00000000  0000  RDCR=0                                  ; READ CSR
        00000002  0000  RDBUF=2                                 ; READ BUFFER
        00000004  0000  OUTCR=4                                 ; OUTPUT CSR
        00000006  0000  OUTB=6                                  ; OUTPUT BUFFER
                  0000
        0000005C  0000  BSLSH=92                                ; BACK SLASH CODE
        0000000D  0000  CR=13                                   ; CARRIAGE RETURN
        0000000A  0000  LF=10                                   ; LINE FEED
        00000027  0000  QUOT=39                                 ; QUOTE
        0000007F  0000  RUBOUT=127                              ; RUBOUT CODE
        0000002F  0000  SLSH=47                                 ; SLASH CODE
                  0000
```

```
              0000
              0000
              0000      ;
              0000      ;       OWN STORAGE:
              0000      ;
          00000000              .PSECT  Z$DEBUG_CODE,LONG,PIC,EXE,WRT
              0000
              0000              .IF     DF,SW_PROCESS
              0000      DELBASE:.LONG   DELBASE-DELBASE              ; RELATIVE PAGE NUMBER OF WRITABLE
              0000              .LONG   <511+DELEND-DELBASE>&^C511; REL PAGE NUMBER OF END OF WRITABLE
              0000              .LONG   DELTA_START-DELBASE          ; START ADDRESS
              0000              .ENDC
              0000
              0000      CONTEXT:                                     ;
 00000000     0000              .LONG   0                            ; BUFFER PADDING
 00000054     0004      INBUF:  .BLKB   80                           ; INPUT BUFFER
 00000000     0054      STATUS: .LONG   0                            ; STATUS FLAGS
 00000000     0058      F1:     .LONG   0                            ; FIELDS
 00000000     005C      F2:     .LONG   0                            ; 1-5
 00000000     0060      F3:     .LONG   0                            ;
 00000000     0064      F4:     .LONG   0                            ;
 00000000     0068      F5:     .LONG   0                            ;
              006C
 00000000     006C      MFYFLG: .LONG   0                            ; MODIFY ENABLE FLAG FOR OTHER PROCESS
              0070                                                   ; ADDRESS SPACES
 00000000     0070      PID:    .LONG   0                            ; PID FOR ADDRESS SPACE 0=>SELF
 00000000     0074      INSLEN: .LONG   0                            ; LENGTH OF PREVIOUS INSTRUCTION
 00000000     0078      INSBUF: .LONG   0                            ; ADDRESS OF INSTRUCTION STREAM BUFFER
              007C                                                   ; (FOR OUTPUT ADDRESS ROUTINE)
       00     007C      FCTR:   .BYTE   0                            ; FIELD COUNTER
              007D
       02     007D      DTYPE:  .BYTE   2                            ; DATA TYPE
       02     007E      CURTYPE:.BYTE   2                            ; CURRENT TYPE
              007F
       00     007F      OPER:   .BYTE   0                            ; OPERATOR
              0080      B:                                           ; BASE OF DATA AREA(CENTER)
 00000000     0080      CURDOT: .LONG   0                            ; CURRENT LOCATION
 00000000     0084      QUAN:   .LONG   0                            ; QUANTITY (;Q)
 00000098     0088      OUTBUF: .BLKL   4                            ; OUTPUT BUFFER
              0098      ;
              0098      ;       REGISTER SAVE AREA
              0098      ;
              0098      SAVREG:                                      ; REGISTER SAVE AREA
 0000009C     0098              .BLKL   1                            ; R0
 000000A0     009C              .BLKL   1                            ; R1
 000000A4     00A0      SAVR2:  .BLKL   1                            ; R2
 000000A8     00A4              .BLKL   1                            ; R3
 000000AC     00A8              .BLKL   1                            ; R4
 000000B0     00AC              .BLKL   1                            ; R5
 000000B4     00B0              .BLKL   1                            ; R6
 000000B8     00B4              .BLKL   1                            ; R7
 000000BC     00B8              .BLKL   1                            ; R8
 000000C0     00BC              .BLKL   1                            ; R9
 000000C4     00C0              .BLKL   1                            ; R10
 000000C8     00C4              .BLKL   1                            ; R11
 000000CC     00C8      SAVAP:  .BLKL   1                            ; AP
 000000D0     00CC              .BLKL   1                            ; (FP)
```

```
000000D4  00D0  SAVSP:  .BLKL   1                           ; SP
000000D8  00D4  SAVPC:  .BLKL   1                           ; PC
000000DC  00D8  SAVPSL: .BLKL   1                           ; PSL
000000DE  00DC  SAVOCR: .BLKW   1                           ; OUTPUT CSR SAVE
000000E0  00DE  SAVRCR: .BLKW   1                           ; INPUT CSR SAVE
          00E0  ASTEN:                                      ; AST ENABLE SAVE LOCATION
000000E4  00E0  SAVRXCS:.BLKL   1                           ; CONSOLE RECEIVER STATUS
          00E4
000000E4  00E4  CONTEXTSZ=.-CONTEXT                         ; SIZE OF PER MODE CONTEXT AREA
          00E4  ;
          00E4  ;       RESERVE SPACE FOR MULTIPLE MODE CONTEXT AREA
          00E4  ;
          00E4          .IF     DF,SW_PROCESS               ;
          00E4          .REPT   3                           ;
          00E4          .BLKB   CONTEXTSZ                   ; FOR EXEC,SUPER AND USER
          00E4  SAV...= .
          00E4          .=.-CONTEXTSZ+<DTYPE-CONTEXT>       ; POINT AT DTYPE,CURTYP
          00E4          .BYTE   2,2                         ; SET TYPE TO LONGWORD
          00E4          .=SAV...                            ; RESTORE LOCATION COUNTER
          00E4          .ENDR                               ;
          00E4          .ENDC                               ;
          00E4
          00E4
          00E4  ;
          00E4  ;       BREAK POINT DATA
          00E4  ;
          00E4
      00  00E4  OVROPC: .BYTE   0                           ; OPCODE IN STEP-OVER BREAKPOINT
          00E5          .ALIGN  LONG
          00E8
000000E4  00E8  BRKADR=.-4
          00E8          .IF     NDF,SW_PROCESS
          00E8  XDELIBRK::
00000000' 00E8          .LONG   INI$BRK                     ; ADDRESS OF INITIAL BREAKPOINT
          00EC          .IFF                                ; FOR PROCESS VERSION
          00EC  INIBRKA:.LONG   0                           ; INITIAL BREAKPOINT
          00EC          .ENDC                               ;
00000108  00EC          .BLKL   7                           ; OTHER BREAK POINT ADDRESSES
00000008  0108  NBRK=<.-4-BRKADR>/4                         ; NUMBER OF BREAKPOINTS
0000010C  0108  OVRADR: .BLKL   1                           ; TEMPORARY BREAKPOINT FOR STEP-OVER
00000001  010C  NTMPBRK=1                                   ; NUMBER OF TEMPORARY BREAKPOINTS
0000010B  010C  BRKOP=.-1                                   ; SAVED OPCODE
      01  010C          NOP                                 ; INITIAL OPCODE
00000114  010D          .BLKB   7                           ; REMAINING OPCODES
00000115  0114          .BLKB   1                           ; TEMPORARY BREAKPOINT OPCODE
          0115
00000111  0115  BRKDSP=.-4
00000135  0115          .BLKL   8                           ; DISPLAY LOCATION START
00000131  0135  BRKCOM=.-4
00000155  0135          .BLKL   8                           ; COMMAND START
          0155
00000161  0155  XREGV:  .BLKL   3                           ; X REGISTER VECTOR
          0161          .IF     NDF,SW_PROCESS              ;
          0161  XDEL_LOADBASE::                             ; BASE OF LOADABLE CPU DEPENDANT CODE
00000000  0161          .LONG   0                           ; X3 = BASE OF SYSLOA CODE
00000000' 0165          .LONG   SCH$GL_CURPCB               ; X4 = CURRENT PCB ADDRESS
00000000' 0169          .LONG   SCH$GL_PCBVEC               ; X5 = BASE OF PCB VECTOR
```

```
00000000' 016D          .LONG   PFN$AW_SWPVBN           ; X6 = SWAP VBN
00000000' 0171          .LONG   PFN$AL_PTE              ; X7 = PTE BACK POINTER
00000000' 0175          .LONG   PFN$AL_BAK              ; X8 = BACKUP ADDRESS
00000000' 0179          .LONG   PFN$AW_REFCNT           ; X9 = REFERENCE COUNT
00000000' 017D          .LONG   PFN$Ax_FLINK            ; XA = FORWARD LINK
00000000' 0181          .LONG   PFN$Ax_BLINK            ; XB = BACK LINK
00000000' 0185          .LONG   PFN$AB_STATE            ; XC = STATE
00000000' 0189          .LONG   PFN$AB_TYPE             ; XD = TYPE
          018D  XDS$GL_XESTRING::
00000000' 018D          .LONG   XDS$GT_WORD_PFN         ; XE;E WITH X0 = PFN , DEFAULT TO WORD ARRAY
          0191  XDS$GL_XFSTRING::
00000000' 0191          .LONG   XDS$GT_WORD_PFN         ; XF;E WITH R0 = PFN , DEFAULT TO WORD ARRAY
00000199  0195  MCHKSAV:.BLKL   1                       ; SAVED CONTENT OF MACHINE CHECK VECTOR
          0199          .IFF                            ; FOR PROCESS VERSION
          0199          .BLKL   13                      ;
          0199  TTIOSB: .BLKL   2                       ; IO STATUS BLOCK FOR TERMINAL READ
          0199  TTCHAN: .BLKL   1                       ; CHANNEL NUMBER
          0199  TTNAMD: .LONG   2                       ; DESCRIPTOR OF INPUT/OUTPUT DEVICE
          0199          .BLKL   1 ;TTNAMD+8             ; (ADDRESS SET BY INITIALIZATION)
          0199          .ASCII  /TT/                    ;
          0199  TTITMLST:                               ; THE ITEMLIST TO ALLOW DELTA TO TURN OFF ED
          0199          .WORD   0
          0199          .WORD   TRM$_MODIFIERS          ; SPECIFY THE MODIFIERS
          0199          .LONG   TRM$M_TM_NOEDIT         ; SPECIFY NO EDITING
          0199          .LONG   0
          0199          .WORD   TERMASKLEN              ; LENGTH OF TERMINATOR MASK
          0199          .WORD   TRM$_TERM               ;SPECIFY THE TERMINATOR MASK
          0199  TERMASKADR:                             ; ALLOW FOR RELOCATION
          0199          .BLKL   2
          0199  TTITMLSTLEN=.-TTITMLST
          0199  DBGINPUT:
          0199          .LONG   9                       ; DESCRIPTOR OF DEFAULT INPUT/OUTPUT
          0199          .BLKL   1 ;DBGINPUT+8
          0199          .ASCII  /DBG$DELTA/             ; FIRST DEFAULT DELTA INPUT
          0199  TRNINPUT:
          0199          .LONG   64                      ; TRANSLATED DBG$DELTA
          0199          .BLKL   1 ;TRNINPUT+8           ; (ADDRESS SET BY INITIALIZATION)
          0199          .BLKB   64
          0199  DBGACTIVE:                              ; ACTIVE FLAGS BY ACCESS MODE
          0199          .LONG   0                       ;
          0199  EXITBLK:                                ; EXIT HANDLER BLOCK
          0199          .LONG   0                       ;
          0199  EXIHADR:.BLKL   1 ;EXIHANDLE            ; EXIT HANDLER (ADDRESS SET BY INIT)
          0199          .LONG   1                       ; ARGUMENT COUNT
          0199  EXCODA: .BLKL   1 ;EXITCODE             ; ADDRESS TO STORE STATUS (ADDRESS SET BY IN
          0199  EXITCODE:                               ;
          0199          .LONG   1                       ; RECEIVER FOR EXIT CODE
          0199  KCOND_PRIMARY:
          0199          .LONG   0                       ; PREVIOUS KERNEL PRIMARY HANDLER
          0199  ECOND_PRIMARY:
          0199          .LONG   0                       ; PREVIOUS EXEC PRIMARY HANDLER
          0199  SCOND_PRIMARY:
          0199          .LONG   0                       ; PREVIOUS SUPER PRIMARY HANDLER
          0199  KCOND_LASTCHANC:
          0199          .LONG   0                       ; PREVIOUS KERNEL LAST CHANCE HANDLER
          0199  ECOND_LASTCHANC:
          0199          .LONG   0                       ; PREVIOUS EXEC LAST CHANCE HANDLER
```

```
          0199  SCOND_LASTCHANC:
          0199          .LONG   0                               ; PREVIOUS SUPER LAST CHANCE HANDLER
          0199  TERMASK:                                        ; TERMINATOR MASK DESCRIPTOR
          0199          .LONG   <1@9>!<1@10>!<1@13>!<1@27>      ; TAB,LF,CR,ESC
          0199          .LONG   <1@1>!<1@2>!<1@15>!<1@29>       ; '!', '"', '/', '='
          0199          .LONG   <1@15>!<1@19>                   ; 'O', 'S'
          0199          .LONG   0                               ;
          0199  TERMASKLEN=.-TERMASK
          0199          .ENDC                                   ;
          0199  ;
          0199  ; LIST OF OPCODES WHICH CALL ROUTINES
          0199  ;
          0199  OVEROPCODES:
      10  0199          .BYTE   ^X10                            ; BSBB
      16  019A          .BYTE   ^X16                            ; JSB
      30  019B          .BYTE   ^X30                            ; BSBW
      FA  019C          .BYTE   ^XFA                            ; CALLG
      FB  019D          .BYTE   ^XFB                            ; CALLS
00000005  019E  OVEROPCLEN = .-OVEROPCODES
```

```
                                      019E          .SBTTL  PRIMARY COMMAND CHARACTER SWITCH
                                      019E
                                      019E      ;
                                      019E      ;       PRIMARY CHARACTER LIST
                                      019E      ;
                                      019E      PRIMARY:                                ;
42 41 39 38 37 36 35 34 33 32 31 30   019E          .ASCII  /0123456789ABCDEF/          ; DECIMAL AND HEX CHARS
                        46 45 44 43   01AA
                                 2E   01AE          .ASCII  /./                         ; DOT - CURRENT LOCATION
                                 2C   01AF          .ASCII  /,/                         ; COMMA - FIELD SEPARATOR
                           00000012   01B0      OPERBAS=.-PRIMARY                       ; OPERATORS
                                 2B   01B0          .ASCII  /+/                         ; PLUS - ADD
                                 20   01B1          .ASCII  / /                         ; BLANK - SAME AS PLUS
                                 40   01B2          .ASCII  /@/                         ; SHIFT OPERATOR
                                 2A   01B3          .ASCII  /*/                         ; MULTIPLY OPERATOR
                                 25   01B4          .ASCII  /%/                         ; DIVIDE OPERATOR
                                 2D   01B5          .ASCII  /-/                         ; MINUS - SUBTRACT OPERATOR
                                 5B   01B6          .ASCII  /[/                         ; LBRACKET - LEFT BRACKET
                                      01B7      TERM:                                   ; BASE OF TERMINATOR LIST
                                 09   01B7          .ASCII  <9>                         ; TAB - INDIRECT
                                 0A   01B8          .ASCII  <10>                        ; LINEFEED -
                                 0D   01B9          .ASCII  <CR>                        ; RETURN -
                                 2F   01BA          .ASCII  '/'                         ; SLASH - OPEN FOR DISPLAY
                                 22   01BB          .ASCII  '"'                         ; DOUBLE QUOTE - OPEN FOR ASCII DISPLAY
                                 3D   01BC          .ASCII  /=/                         ; EQUALS - DISPLAY
                                 1B   01BD          .ASCII  <27>                        ; ESCAPE - PREVIOUS LOCATION
                                 53   01BE          .ASCII  /S/                         ; STEP
                                 4F   01BF          .ASCII  /O/                         ; STEP-OVER ROUTINE
                                 21   01C0          .ASCII  /!/                         ; DISPLAY INSTRUCTION
                           0000000A   01C1      NTERM=.-TERM                            ; NUMBER OF TERMINATORS
                                 3B   01C1          .ASCII  <59>                        ; SEMI - INITIATE SECONDARY
                                 3A   01C2          .ASCII  /:/                         ; COLON - SEPARATE PID FORM ADDRESS
                                 50   01C3          .ASCII  /P/                         ; P - PROCESSOR REGISTER PREFIX
                                 51   01C4          .ASCII  /Q/                         ; Q - LAST QUANTITY
                                 27   01C5          .ASCII  /'/                         ; QUOTE - BEGIN CHAR STRING
                                 52   01C6          .ASCII  /R/                         ; REGISTER PREFIX
                                 47   01C7          .ASCII  /G/                         ; G - GLOBAL PREFIX
                                 48   01C8          .ASCII  /H/                         ; H - HIGH, P1 SPACE PREFIX
                                 58   01C9          .ASCII  /X/                         ; X REGISTER PREFIX
                           0000002C   01CA      NPRIM=.-PRIMARY                         ; NUMBER OF PRIMARY COMMANDS
                                      01CA
```

```
                            01CA          .SBTTL  PRIMARY COMMAND SCANNER
                            01CA
                            01CA   ;
                            01CA   ;      PRIMARY COMMAND SCANNER
                            01CA   ;
                            01CA
                            01CA
00 0D 0A 3F 48 45 0D 0A     01CA   OUTER:  .ASCIZ  <LF><CR>/EH?/<LF><CR>
                            01D2
                      0000  01D2   DCOM:   .WORD                        ; CALL ENTRY POINT
                            01D4           .IF     DF,SW_PROCESS        ; FOR PROCESS VERSION ONLY
                            01D4           MOVAB   W^DBGEXCEP,(FP)      ; SET CONDITION HANDLER ADDRESS
                            01D4           .ENDC
               13      11   01D4           BRB     SCANP                ; ENTER SCANP
         54    F1 AF   9E   01D6   ERROR:  MOVAB   OUTER,R4             ; SET ADDR OF CONTROL STRING
                 0344  30   01DA           BSBW    OUTZSTRING           ; OUTPUT ASCIZ STRING
               5E  5D  D0   01DD   SUPERST:MOVL    FP,SP                ; RESET STACK
         59    84 AB   9E   01E0           MOVAB   INBUF-B(R11),R9      ; RESET STRING ADDRESS
                   69  94   01E4           CLRB    (R9)                 ; AND FORCE READ
                 045A  30   01E6           BSBW    RESET                ; RESET SCANNER
                   02  10   01E9   SCANP:  BSBB    NEXTP                ; SCAN INPUT
                   FC  11   01EB           BRB     SCANP                ; SCAN IT ALL
                            01ED   NEXTP:                               ; PROCESS NEXT PRIMARY CHAR
                 036A  30   01ED           BSBW    GETCHAR              ; GET CHARACTER
   A9 AF   2C      58  3A   01F0           LOCC    R8,#NPRIM,PRIMARY    ; CHECK IT
                   DF  13   01F5           BEQL    ERROR                ; NOT FOUND, ERROR
      50   2C      50  C3   01F7           SUBL3   R0,#NPRIM,R0         ; RATIONALIZE INDEX
                            01FB           CASE    R0,LIMIT=#16,<-
                            01FB                   DOT,-                ; DOT - CURRENT LOCATION
                            01FB                   COMMA,-              ; COMMA - FIELD SEPARATOR
                            01FB                   OPERATOR,-           ; PLUS - ADD OPERATOR
                            01FB                   OPERATOR,-           ; BLANK - ADD OPERATOR
                            01FB                   OPERATOR,-           ; @ - SHIFT OPERATOR
                            01FB                   OPERATOR,-           ; * - MULTIPLY OPERATOR
                            01FB                   OPERATOR,-           ; % - DIVIDE OPERATOR
                            01FB                   NEGATE,-             ; MINUS - SUBTRACT/NEGATE
                            01FB                   LBRACKET,-           ; LEFT BRACKET - MODE SELECT
                            01FB                   TAB,-                ; TAB - INDIRECT
                            01FB                   LINEFEED,-           ; LINE FEED - NEXT LOCATION
                            01FB                   RETURN,-             ; RETURN - CLOSE OPEN CELL
                            01FB                   SLASH,-              ; SLASH - OPEN FOR DISPLAY
                            01FB                   DQUOTE,-             ; DOUBLE QUOTE - OPEN FOR ASCII DISPLAY
                            01FB                   EQUALS,-             ; EQUALS - DISPLAY VALUE
                            01FB                   ESCAP,-              ; ESCAPE - PREVIOUS LOCATION
                            01FB                   STEP,-               ; 'S' - SINGLE STEP
                            01FB                   STEPOVER,-           ; 'O' - STEP OVER ROUTINE CALL
                            01FB                   INSTR,-              ; '!' - DISPLAY INSTRUCTION
                            01FB                   SEMI,-               ; SEMI COLON - SECONDARY COMMAND
                            01FB                   COLON,-              ; COLON - SEPARATE PID FROM ADDRESS
                            01FB                   PREG,-               ; 'P' - PROCESSOR REGISTER
                            01FB                   QUANT,-              ; 'Q' - LAST QUANTITY
                            01FB                   QUOTE,-              ; QUOTE - BEGIN ASCII STRING
                            01FB                   REGISTER,-
                            01FB                   GLOBL,-              ; G - GLOBAL PREFIX
                            01FB                   HIGH,-               ; H - P1 SPACE PREFIX
                            01FB                   XREG,-               ; X REGISTER
                            01FB                   >                    ;
```

```
           10    50  B1 0237         CMPW    R0,#16                    ; IS NUMBER > RADIX
                 9A  18 023A         BGEQ    ERROR                     ; YES
           56    10  C4 023C         MULL    #16,R6                    ; SCALE BY RADIX
           56    50  C0 023F         ADDL    R0,R6                     ; AND ADD NEW DIGIT
           6A    04  C8 0242 INFLD:  BISL    #<1@V_INFIELD>,(R10)      ; NOTE FIELD INPUT
                     05 0245         RSB                               ; NEXT PRIMARY CHARACTER
                        0246
                        0246
      54   01    1F  9C 0246 GLOBL:  ROTL    #31,#1,R4                 ; GENERATE SYSTEM SPACE PREFIX
                 07  11 024A         BRB     PRE1                      ; MERGE WITH COMMON
  54   7FFE0000  8F  D0 024C HIGH:   MOVL    #^X7FFE0000,R4            ; P1 SPACE BASE ADDRESS
                 06  10 0253 PRE1:   BSBB    ENDEXPR                   ; END EXPRESSION
           56    54  D0 0255         MOVL    R4,R6                     ; SET INTO ACCUM
              E7 AF  9F 0258         PUSHAB  INFLD                     ; RETURN THROUGH INFLD
                        025B ;       BRB     ENDEXPR
                        025B
```

```
                        025B          .SBTTL  ENDEXPR - END EXPRESSION
                        025B
                        025B      ;
                        025B      ;
                        025B      ;
                        025B      ENDEXPR:                                    ;
        03 6A   07  E5  025B              BBCC    #V_NEGATE,(R10),5$          ; SKIP IF NOT NEGATE
           56   56  CE  025F              MNEGL   R6,R6                       ; NEGATE ACCUMULATOR
                06  10  0262      5$:     BSBB    10$                         ; PERFORM OPERATION
                56  D4  0264              CLRL    R6                          ; CLEAR ACCUMULATOR
             FF AB  94  0266              CLRB    OPER-B(R11)                 ; INIT OPERATOR
                    05  0269              RSB                                 ; AND RETURN
                        026A      10$:    CASE    OPER-B(R11),TYPE=B,<-       ; DO OPERATION
                        026A                      ADD,-                       ; ADD, PLUS
                        026A                      ADD,-                       ; BLANK, PLUS
                        026A                      SHFT,-                      ; SHIFT, @
                        026A                      MUL,-                       ; MULTIPLY, *
                        026A                      DIV,-                       ; DIVIDE, %
                        026A                      >                           ;
   57   57   56     78  0279      SHFT:   ASHL    R6,R7,R7                    ; SHIFT
                    05  027D              RSB                                 ; AND EXIT
        57   56     C4  027E      MUL:    MULL    R6,R7   ; MULTIPLY
                    05  0281              RSB                                 ; AND EXIT
        57   56     C6  0282      DIV:    DIVL    R6,R7   ; DIVIDE
                    05  0285              RSB                                 ; AND EXIT
        57   56     C0  0286      ADD:    ADDL    R6,R7   ; ADD
                    05  0289              RSB                                 ; AND EXIT
                        028A
```

```
                        028A        .SBTTL  SLASH - OPEN CELL
                        028A
                        028A  ;
                        028A  ;     OPEN SPECIFIED CELL
                        028A  ;
                        028A  DQUOTE:                                             ; DISPLAY IN ASCII
          6A   02   88  028A        BISB    #<1@V_ASCII>,(R10)                  ; SET ASCII FLAG
               05   11  028D        BRB     OPEN                                ;
                        028F
                        028F  SLASH:                                              ;
     6A   2002 8F   AA  028F        BICW    #<1@V_ASCII>!<1@V_INSTR>,(R10)   ; CLEAR ASCII DISPLAY MODE
     6A   2000 8F   AA  0294  OPEN: BICW    #<1@V_INSTR>,(R10)                  ; CLEAR INSTRUCTION FLAG
               4E   10  0299        BSBB    ENDFIELD                            ; TERMINATE FIELD
        06 6A  08   E0  029B        BBS     #V_F1,(R10),5$                      ; ADDR SPECIFIED?
        6B   04 AB  D0  029F        MOVL    QUAN-B(R11),CURDOT-B(R11)           ; NO, GO INDIRECT
               04   11  02A3        BRB     10$                                 ; AND DISPLAY CONTENT
        6B   D8 AB  D0  02A5  5$:   MOVL    F1-B(R11),CURDOT-B(R11)             ; SET NEW DOT
  50  6A  01   0F   EF  02A9  10$:  EXTZV   #V_PRMODE,#1,(R10),R0               ; GET PROCESSOR REGISTER MODE FLAG
  6A  01  1F   50   F0  02AE        INSV    R0,#V_PREG,#1,(R10)                 ; AND MOVE TO SEMI-PERMANENT COPY
             0098   30  02B3        BSBW    LOCOUT                              ; OUTPUT AND OPEN
        22 6A  09   E1  02B6        BBC     #V_F2,(R10),RSET                    ; RANGE SPECIFIED?
        6B   DC AB  D1  02BA  15$:  CMPL    F2-B(R11),CURDOT-B(R11)             ; CHECK FOR END
               1C   15  02BE        BLEQ    RSET                                ; YES
             0086   30  02C0        BSBW    NEXTLOC                             ; INCREMENT TO NEXT DOT
               F5   11  02C3        BRB     15$                                 ; AND CONTINUE
             FF0E   31  02C5  ERR4: BRW     ERROR                               ; DECLARE ERROR
                        02C8
```

```
                          02C8      .SBTTL  RETURN - CLOSE CURRENT OPEN CELL
                          02C8
                          02C8  ;
                          02C8  ;       RETURN - CLOSE CURRENT OPEN CELL
                          02C8  ;
                          02C8
                          02C8  RETURN:                                 ;
           1F      10     02C8          BSBB    ENDFIELD                ; TERMINATE CURRENT FIELD
                          02CA          .ENABL  LSB                     ;
     11 6A    00   E5     02CA          BBCC    #V_OPEN,(R10),10$       ; SKIP IF NONE OPEN
  6A   2002 8F     B3     02CE          BITW    #<1@V_ASCII>!<1@V_INSTR>,(R10)  ; IF ASCII OR INSTRUCTION
           07      12     02D3          BNEQ    RSET                    ; DISPLAY MODE SKIP STORE OPERATION
     03 6A    08   E1     02D5          BBC     #V_F1,(R10),RSET        ; SKIP IF NOTHING TO STORE
         07AB      30     02D9          BSBW    DEPOSIT                 ; DEPOSIT
         0364      31     02DC  RSET:   BRW     RESET                   ; RESET SCANNER
     F9 6A    08   E1     02DF  10$:    BBC     #V_F1,(R10),RSET        ; DONE IF NO INPUT
         0355      31     02E3          BRW     EQ[1                    ; OTHERWISE OUTPUT
                          02E6          .DSABL  LSB                     ;
```

```
                    02E6         .SBTTL  ENDFIELD - TERMINATE CURRENT FIELD
                    02E6
                    02E6  ;
                    02E6  ;      COMMA TERMINATE CURRENT FIELD
                    02E6  ;
       FF59    30   02E6  COMMA:  BSBW    INFLD                      ; ZERO IF NULL FIELD
                    02E9
                    02E9  ;
                    02E9  ;      TERMINATE CURRENT FIELD
                    02E9  ;
                    02E9  ENDFIELD:                                  ;
    16 6A   02 E5   02E9          BBCC    #V_INFIELD,(R10),10$       ; CLEAR PENDING FIELD
       FF6B    30   02ED          BSBW    ENDEXPR                    ; END EXPRESSION
    50  FC AB  9A   02F0          MOVZBL  FCTR-B(R11),R0             ; GET FIELD POINTER
 CC 01 AA   50 E2   02F4          BBSS    R0,1(R10),ERR4             ; ERROR IF TOO MANY FIELDS
  D8 AB40   57 D0   02F9          MOVL    R7,F1-B(R11)[R0]           ; STORE FIELD VALUE
       FC AB   96   02FE          INCB    FCTR-B(R11)                ; INCREMENT FIELD COUNTER
          56   7C   0301          CLRQ    R6                         ; CLEAR ACCUMULATORS
               05   0303  10$:    RSB                                ; RETURN
                    0304
```

```
                     0304        .SBTTL  FETCH - OBTAIN DATA SPECIFIED
                     0304
                     0304  ;
                     0304  ;     FETCH SPECIFIED DATA
                     0304  ;
                     0304  FETCH:                                      ;
      1D 6A  1F   E0 0304        BBS     #V_PREG,(R10),40$           ; BR IF PROCESSOR REGISTER
                     0308        .IF     DF,SW_PROCESS
                     0308        TSTL    PID-B(R11)                  ; CHECK FOR PROCESS GET
                     0308        BNEQ    50$                         ; BR IF YES
                     0308        .ENDC
                     0308        CASE    CURTYPE-B(R11),TYPE=B,<-         ; OPERATE ON TYPE
                     0308                10$,-                       ; BYTE
                     0308                20$,-                       ; WORD
                     0308                30$,-                       ; LONG
                     0308                >
   04 AB   00 BB  9A 0313  10$:  MOVZBL  @CURDOT-B(R11),QUAN-B(R11)      ; GET BYTE
                  05 0318        RSB                                 ; RETURN
   04 AB   00 BB  3C 0319  20$:  MOVZWL  @CURDOT-B(R11),QUAN-B(R11)      ; GET WORD
                  05 031E        RSB                                 ; RETURN
   04 AB   00 BB  D0 031F  30$:  MOVL    @CURDOT-B(R11),QUAN-B(R11)      ; GET LONGWORD
                  05 0324        RSB                                 ; RETURN
                     0325        .IF     NDF,SW_PROCESS
      04 AB    6B DB 0325  40$:  MFPR    CURDOT-B(R11),QUAN-B(R11)       ; GET PROCESSOR REGISTER
                  05 0329        RSB
                     032A        .IFF                                ; FALSE IF PROCESS VERSION
                     032A  40$:
                     032A        $CMKRNL_S       B^FTCHPREG,(AP)     ; CALL IN KERNEL MODE TO FETCH
                     032A        RSB                                 ;
                     032A  50$:  BRW     FETCHP                      ; FETCH FROM FOREIGN PROCESS
                     032A        .ENDC
                     032A
                     032A        .IF     DF,SW_PROCESS               ;
                     032A  FTCHPREG:
                     032A        .WORD   0                           ; ENTRY MASK
                     032A        MOVAB   W^PREXC,(FP)                ; SET EXCEPTION HANDLER
                     032A        MFPR    CURDOT-B(R11),QUAN-B(R11)       ; GET PROCESSOR REGISTER
                     032A        MOVL    #1,R0                       ; RETURN SUCCESS
                     032A        RET                                 ;
                     032A
                     032A        .ENDC                               ;
```

```
                          032A         .SBTTL  NEXTDOT - INCREMENT CURRENT LOCATION
                          032A
                          032A      ;
                          032A      ;       INCREMENT TO NEXT LOCATION
                          032A      ;
                          032A      NEXTDOT:                                        ;
      10 6A    0D    E0   032A              BBS     #V_INSTR,(R10),20$              ; BRANCH IF INSTRUCTION MODE
         51    01    D0   032E              MOVL    #1,R1                           ; ASSUME UNIT INCREMENT
               6A    D5   0331              TSTL    (R10)                           ; CHECK FOR PREG
               05    19   0333              BLSS    10$                             ; YES, USE UNIT INCREMENT
   51    51 FE AB    9C   0335              ROTL    CURTYPE-B(R11),R1,R1            ; FORM INCREMENT
         6B    51    C0   033A      10$:    ADDL    R1,CURDOT-B(R11)                ; AND ADD TO DOT
                     05   033D              RSB                                     ; RETURN
      6B    F4 AB    C0   033E      20$:    ADDL    INSLEN-B(R11),CURDOT-B(R11) ; SKIP OVER PREVIOUS INSTRUCTION
                     05   0342              RSB
                          0343
```

```
                         0343             .SBTTL  OUTPUT - DISPLAY CONTENT
                         0343     ;
                         0343     ;       OUTPUT CONTENT
                         0343     ;
                         0343     OUTBB:
            1C 0C 04     0343             .BYTE   4,12,28                   ; STARTING DIGIT LIST
                         0346
                         0346             .SBTTL  LINE FEED - DISPLAY NEXT
                         0346
                         0346     LINEFEED:                                 ;
             FF7F  30    0346             BSBW    RETURN                    ; CLOSE OPEN CELL
                         0349     NEXTLOC:                                  ; PROMPT WITH NEXT LOCATION
               DF  10    0349             BSBB    NEXTDOT                   ; INCREMENT LOCATION
                         034B     LOCPROMPT:                                ; DISPLAY ADDR/CONTENT
             0154  30    034B             BSBW    OUTPUTA                   ; OUTPUT ADDRESS
        31 6A  0D  E0    034E     LOCOUT: BBS     #V_INSTR,(R10),OUTINS     ; BRANCH IF INSTRUCTION MODE
               B0  10    0352             BSBB    FETCH                     ; FETCH CONTENT
           6A  01  88    0354             BISB    #<1@V_OPEN>,(R10)         ; INDICATE OPEN CELL
                         0357
                         0357     OUTPUT:                                   ;
        51  FE AB  9A    0357             MOVZBL  CURTYPE-B(R11),R1         ; GET TYPE
      52  E4 AF41  9A    035B             MOVZBL  OUTBB[R1],R2              ; INIT DIGIT SELECTOR
        53  04 AB  D0    0360             MOVL    QUAN-B(R11),R3            ; GET QUANTITY TO DISPLAY
        05 6A  01  E0    0364             BBS     #V_ASCII,(R10),10$        ; CHECK FOR ASCII OUT
             019C  30    0368             BSBW    OUTCOM                    ; OUTPUT NUMBER IN HEX
               0F  11    036B             BRB     20$                       ; AND EXIT THROUGH OUTSPACE
        08 AB  53  D0    036D     10$:    MOVL    R3,OUTBUF-B(R11)          ; PUT STRING IN BUFFER
        52  01 51  78    0371             ASHL    R1,#1,R2                  ; GET COUNT
          08 AB42  94    0375             CLRB    OUTBUF-B(R11)[R2]         ; MARK END OF STRING
             01A1  30    0379             BSBW    OUTZBUF                   ; OUTBUT ASCIIZ BUFFER
             01CC  31    037C     20$:    BRW     OUTSPACE                  ; FOLLOW WITH SPACE
                         037F
```

```
                          037F          .SBTTL  OUTINS - OUTPUT INSTRUCTION
                          037F  ;
                          037F  ;       OUTPUT RANGE OF INSTRUCTIONS
                          037F  ;
           00 09 20 20    037F  SPACES: .ASCIZ  '       '                ; 2 SPACES AND A TAB
                          0383
                          0383          .WEAK   LIB$INS_DECODE          ; INSTRUCTION DECODE IS OPTIONAL
                          0383
  50    00000000'GF   9E  0383  OUTINS: MOVAB   G^LIB$INS_DECODE,R0     ; GET ADDRESS OF INSTRUCTION DECODER
                  07  12  038A          BNEQ    5$                      ; BRANCH IF LINKED WITH DECODER
     6A   2000 8F     AA  038C          BICW    #1@V_INSTR,(R10)        ; SUPPRESS INSTRUCTION MODE
                  BB  11  0391          BRB     LOCODT                  ; AND PRINT 1ST LONGWORD OF INS STREAM
  5E    00000052 8F   C2  0393  5$:     SUBL    #32+50,SP               ; ALLOCATE SPACE FOR INSTRUCTION STREAM
                          039A                                          ; AND DECODE OUTPUT BUFFER
         F8 AB    5E  D0  039A          MOVL    SP,INSBUF-B(R11)        ; SAVE ADDRESS FOR OUTPUT_ADDRESS
              54  08  D0  039E          MOVL    #32/4,R4                ; SET ITERATION COUNT
              55  5E  D0  03A1          MOVL    SP,R5                   ; SET POINTER INTO BUFFER
         FE AB    02  90  03A4          MOVB    #2,CURTYPE-B(R11)       ; SET FOR LONGWORD FETCHES
                  6B  DD  03A8          PUSHL   CURDOT-B(R11)           ; SAVE CURRENT LOCATION COUNTER
                FF57  30  03AA  10$:    BSBW    FETCH                   ; FETCH LONGWORD
         85    04 AB  D0  03AD          MOVL    QUAN-B(R11),(R5)+       ; STORE INTO INSTRUCTION BUFFER
            6B    04  C0  03B1          ADDL    #4,CURDOT-B(R11)        ; SKIP TO NEXT LONGWORD
              F3  54  F5  03B4          SOBGTR  R4,10$                  ; FILL ENTIRE BUFFER
             6B 8ED0      03B7          POPL    CURDOT-B(R11)           ; RESTORE CURRENT LOCATION
                  55  DD  03BA          PUSHL   R5                      ; ADDRESS OF DECODE OUTPUT BUFFER
                  32  DD  03BC          PUSHL   #50                     ; LENGTH OF DECODE OUTPUT BUFFER
               FE'AF  9F  03BE          PUSHAB  B^OUTPUT_ADDRESS        ; ADDRESS OF SYMBOLIZE ROUTINE
               04 AE  3F  03C1          PUSHAW  4(SP)                   ; ADDRESS OF WORD TO RECEIVE LENGTH
               08 AE  7F  03C4          PUSHAQ  8(SP)                   ; ADDRESS OF DECODE OUTPUT DESCRIPTOR
               F8 AB  DF  03C7          PUSHAL  INSBUF-B(R11)           ; ADDRESS OF INSTRUCTION STREAM POINTER
     00000000'GF  04  FB  03CA          CALLS   #4,G^LIB$INS_DECODE     ; DECODE INSTRUCTION INTO BUFFER
              53  8E  7D  03D1          MOVQ    (SP)+,R3                ; GET DESCRIPTOR OF STRING
              1A  50  E9  03D4          BLBC    R0,90$                  ; BRANCH IF ERROR DETECTED
                6443  94  03D7          CLRB    (R4)[R3]                ; MAKE INTO ASCIZ STRING
                0144  30  03DA          BSBW    OUTZSTRING              ; OUTPUT ASCIZ STRING
  F4 AB    F8 AB  5E  C3  03DD          SUBL3   SP,INSBUF-B(R11),INSLEN-B(R11) ; SET LENGTH OF INSTRUCTION
  5E    00000052 8F   C0  03E3  50$:    ADDL    #32+50,SP               ; DEALLOCATE STREAM/DECODE BUFFERS
         54    92 AF  9E  03EA          MOVAB   SPACES,R4               ; SET ADDRESS OF SPACES
                0130  31  03EE          BRW     OUTZSTRING              ; FOLLOW INSTRUCITON WITH SOME SPACE
                          03F1  ;
                          03F1  ; UNABLE TO DECODE INSTRUCTION (ACCVIO OR NEW INSTRUCTION).  OUTPUT LONGWORD
                          03F1  ;
         53    F8 BB  D0  03F1  90$:    MOVL    @INSBUF-B(R11),R3       ; GET FIRST LONGWORD OF STREAM
         F4 AB    01  D0  03F5          MOVL    #1,INSLEN-B(R11)        ; SET INSTRUCTION LENGTH TO 1
                0108  30  03F9          BSBW    OUTLONG                 ; OUTPUT AS LONGWORD
                  E5  11  03FC          BRB     50$
                          03FE
                          03FE  ;
                          03FE  ; OUTPUT AN OPERAND WHICH IS A RELATIVE OR ABSOLUTE ADDRESS
                          03FE  ;
                          03FE
                          03FE  OUTPUT_ADDRESS:
                    081C  03FE          .WORD   ^M<R2,R3,R4,R11>
                          0400
         53    04 BC  D0  0400          MOVL    @4(AP),R3               ; GET VALUE (ARGUMENT BY REFERENCE)
         52    08 AC  D0  0404          MOVL    8(AP),R2                ; GET ADDRESS OF DESCRIPTOR
      0C    10 BC     E8  0408          BLBS    @16(AP),5$              ; BRANCH IF ABSOLUTE ADDRESS
```

```
                          040C         .IF      DF,SW_PROCESS                   ; IF PROCESS VERSION,
                          040C         MOVPSL   R1                              ; GET CURRENT PSL
                          040C         EXTZV    #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1 ; ISOLATE CURRENT ACCESS MODE
                          040C         MULW     #CONTEXTSZ,R1                   ; COMPUTE OFFSET FROM KERNEL CONTEXT
                          040C         MOVAB    W^B[R1],R11                     ; GET BASE ADDRESS OF CONTEXT AREA
                          040C         .IFF                                     ; FOR EXECUTIVE VERSION,
     5B    FC70 CF    9E  040C         MOVAB    W^B,R11                         ; GET BASE ADDRESS OF CONTEXT AREA
                          0411         .ENDC
      53     F8 AB    C2  0411         SUBL     INSBUF-B(R11),R3                ; GET OFFSET FROM INSTRUCTION
          53    6B    C0  0415         ADDL     CURDOT-B(R11),R3                ; AND COMPUTE ''REAL'' ADDRESS
       04 AB    53    D0  0418  5$:    MOVL     R3,QUAN-B(R11)                  ; SET NEW "Q" SO THAT INDIRECTION
                          041C                                                  ; CAN BE USED TO SEE THE LAST OPERAND
                          041C                                                  ; OR BRANCHED-TO INSTRUCTION
           0C BC      B4  041C         CLRW     @12(AP)                         ; ASSUME NOTHING PUT INTO BUFFER
         08     62    B1  041F         CMPW     (R2),#8                         ; ENOUGH ROOM FOR 8 CHARACTERS?
                39    19  0422         BLSS     20$                             ; BRANCH IF NOT ENOUGH
             04 A2    DD  0424         PUSHL    4(R2)                           ; SAVE ADDRESS OF RESULT BUFFER
          51    53    D0  0427         MOVL     R3,R1                           ; COPY EFFECTIVE ADDRESS
              0045    30  042A         BSBW     RELOC                           ; SEE IF CLOSE TO RELOCATION REGISTER
                1F    19  042D         BLSS     8$                              ; BRANCH IF NONE FOUND
          52    54    D0  042F         MOVL     R4,R2                           ; SAVE OFFSET FROM X REGISTER
             54 8ED0      0432         POPL     R4                              ; GET ADDRESS OF OUTPUT BUFFER
     84      58 8F    90  0435         MOVB     #^A'X',(R4)+                    ; WRITE 'X'
                50    D4  0439         CLRL     R0                              ; PRINT ONLY 1 DIGIT
                24    10  043B         BSBB     100$                            ; WRITE REGISTER NUMBER
          84    2B    90  043D         MOVB     #^A'+',(R4)+                    ; WRITE '+'
          50    08    D0  0440         MOVL     #8,R0                           ; PRINT 3 DIGITS
          53    52    D0  0443         MOVL     R2,R3                           ; RETRIEVE OFFSET FROM X REGISTER
                19    10  0446         BSBB     100$                            ; WRITE OFFSET IN HEX
     0C BC      06    B0  0448         MOVW     #6,@12(AP)                      ; STORE LENGTH OF STRING
                0F    11  044C         BRB      20$                             ; EXIT SUCCESSFULLY
          53    51    D0  044E  8$:    MOVL     R1,R3                           ; GET EFFECTIVE ADDRESS
             54 8ED0      0451         POPL     R4                              ; GET ADDRESS OF OUTPUT BUFFER
          50    1C    D0  0454         MOVL     #28,R0                          ; SET STARTING BIT FOR 1ST DIGIT
                08    10  0457         BSBB     100$                            ; OUTPUT HEX LONGWORD
     0C BC      08    B0  0459         MOVW     #8,@12(AP)                      ; RETURN LENGTH TO CALLER
          50    01    D0  045D  20$:   MOVL     #1,R0                           ; SUCCESS
                      04  0460         RET
                          0461
 51   53    04    50  EF  0461  100$:  EXTZV    R0,#4,R3,R1                     ; GET DIGIT
   84    FD33 CF41    90  0466         MOVB     PRIMARY[R1],(R4)+               ; MOVE DIGIT INTO BUFFER
          50    04    C2  046C         SUBL     #4,R0                           ; SKIP TO NEXT DIGIT
                F0    18  046F         BGEQ     100$                            ; LOOP UNTIL END OF LONGWORD
                      05  0471         RSB
```

```
                                 0472           .SBTTL  DETERMINE CLOSEST RELOCATION REGISTER
                                 0472   ;
                                 0472   ; RELOC - GIVEN AN ADDRESS, RETURN CLOSEST RELOCATION REGISTER, IF ANY.
                                 0472   ;
                                 0472   ; INPUTS:
                                 0472   ;
                                 0472   ;       R1 = ADDRESS
                                 0472   ;
                                 0472   ; OUTPUTS:
                                 0472   ;
                                 0472   ;       X1 = EFFECTIVE ADDRESS
                                 0472   ;       R3 = REGISTER #
                                 0472   ;       R4 = OFFSET FROM X REGISTER
                                 0472   ;       PSL CONDITION CODES SET ON R3
                                 0472   ;       R2 DESTROYED.
                                 0472   ;
                      53    D4   0472   RELOC:  CLRL    R3                      ; START WITH X0
                 52   01    CE   0474           MNEGL   #1,R2                   ; X REGISTER CLOSEST TO ADDRESS
                 54   52    3C   0477           MOVZWL  R2,R4                   ; CLOSEST SO FAR IS FFFF
      50    FCD6 CF43       D0   047A   10$:    MOVL    XREGV[R3],R0            ; GET X REGISTER
                      18    13   0480           BEQL    15$                     ; BRANCH IF NOT VALID
           50    51   50    C3   0482           SUBL3   R0,R1,R0                ; GET OFFSET FROM X#
       00000800  8F   50    D1   0486           CMPL    R0,#^X800               ; WITHIN REASONABLE RANGE?
                      0B    1E   048D           BGEQU   15$                     ; BRANCH IF OK
                 54   50    D1   048F           CMPL    R0,R4                   ; CLOSER THAN CLOSEST SO FAR?
                      06    1A   0492           BGTRU   15$                     ; BRANCH IF NOT
                 52   53    D0   0494           MOVL    R3,R2                   ; SAVE X# CLOSEST TO ADDRESS
                 54   50    D0   0497           MOVL    R0,R4                   ; AND SET NEW CLOSEST OFFSET
              DC 53   08    F2   049A   15$:    AOBLSS  #8,R3,10$               ; LOOP UNTIL LAST REGISTER TESTED
                 53   52    D0   049E           MOVL    R2,R3                   ; RETURN X# CLOSEST TO ADDRESS
                            05   04A1           RSB
```

```
                      04A2          .SBTTL  OUTPUTA - OUTPUT ADDRESS
                      04A2  ;
                      04A2  ;       OUTPUT ADDRESS
                      04A2  ;
                      04A2  OUTPUTA:                                  ; OUTPUT ADDRESS
         00AB   30    04A2          BSBW    CRLF                      ; OUTPUT CR/LF
      51   6B   D0    04A5          MOVL    CURDOT-B(R11),R1          ; GET ADDRESS
           C8   10    04A8          BSBB    RELOC                     ; SEE IF CLOSE TO RELOCATION REGISTER
           1B   19    04AA          BLSS    2$                        ; BRANCH IF NOT
           54   DD    04AC          PUSHL   R4                        ; SAVE OFFSET FROM RELOCATION REGISTER
   50   58 8F   9A    04AE          MOVZBL  #^A'X',R0                 ; OUTPUT AN 'X'
         007F   30    04B2          BSBW    OUTCHAR
           52   D4    04B5          CLRL    R2                        ; PRINT ONLY 1 HEX DIGIT
           4E   10    04B7          BSBB    OUTCOM                    ; OUTPUT HEX VALUE IN R3
      50   2B   9A    04B9          MOVZBL  #^A'+',R0                 ; OUTPUT AN '+'
         0075   30    04BC          BSBW    OUTCHAR
      52   08   D0    04BF          MOVL    #8,R2                     ; PRINT 3 DIGITS
         53   8ED0    04C2          POPL    R3                        ; GET OFFSET FROM RELCATION REGISTER
           31   11    04C5          BRB     10$                       ; OUTPUT OFFSET AND SLASH
   53   18 AB   9E    04C7  2$:     MOVAB   SAVREG-B(R11),R3          ; BASE OF REGISTER AREA
                      04CB          .IF     DF,SW_PROCESS             ; ONLY FOR PROCESS VERSION
                      04CB          TSTL    PID-B(R11)                ; CHECK FOR OTHER PROCESS ADDRESS
                      04CB          BNEQ    3$                        ; BR IF YES
                      04CB          .ENDC                             ;
53   6B   53    C3    04CB          SUBL3   R3,CURDOT-B(R11),R3       ; COMPUTE OFFSET INTO REGISTER AREA
           13   19    04CF          BLSS    5$                        ; NOT GENERAL REGISTER
      53   04   C6    04D1          DIVL    #4,R3                     ; SCALE TO LONGWORD NUMBER
      0F   53   D1    04D4          CMPL    R3,#15                    ; CHECK FOR MAX REG NUMBER
           0B   14    04D7          BGTR    5$                        ; GTR, NOT A REGISTER
   50   52 8F   9A    04D9          MOVZBL  #^A'R',R0                 ; OUTPUT PREFIX
         0054   30    04DD          BSBW    OUTCHAR                   ; OF 'R'
           52   D4    04E0          CLRL    R2                        ; AND SET FOR ONE DIGIT OF OUTPUT
           14   11    04E2          BRB     10$                       ;
                      04E4          .IF     DF,SW_PROCESS             ; FOR PROCESS VERSION ONLY
                      04E4  3$:     TSTL    (R10)                     ; CHECK FOR PROCESSOR REGISTER
                      04E4          BLSS    5$                        ; BR IF YES
                      04E4          MOVL    #28,R2                    ; SET FOR LONGWORD OUTPUT
                      04E4          MOVL    PID-B(R11),R3             ; GET PID OF TARGET
                      04E4          BSBB    OUTCOM                    ; OUTPUT PID AS LONGWORD
                      04E4          MOVZBL  #^A':',R0                 ; SEPARATE WITH ':'
                      04E4          BSBW    OUTCHAR                   ; OUTPUT COLON
                      04E4          .ENDC                             ;
      53   6B   D0    04E4  5$:     MOVL    CURDOT-B(R11),R3          ; GET ADDRESS
      52   1C   D0    04E7          MOVL    #28,R2                    ; ASSUME LONGWORD OUTPUT
           6A   D5    04EA          TSTL    (R10)                     ; CHECK FOR PROCESSOR REGISTER
           0A   18    04EC          BGEQ    10$                       ; NO, JUST A LONGWORD
   50   50 8F   9A    04EE          MOVZBL  #^A'P',R0                 ; PRECEDE WITH A 'P'
         003F   30    04F2          BSBW    OUTCHAR                   ; OUTPUT P
      52   04   D0    04F5          MOVL    #4,R2                     ; SET FIELD TO 2 DIGITS
           0D   10    04F8  10$:    BSBB    OUTCOM                    ; COMMON OUTPUT
      50   2F   9A    04FA          MOVZBL  #SLSH,R0                  ; OUTPUT SLASH
         0034   31    04FD          BRW     OUTCHAR                   ; RETURN THROUGH OUTCHAR
                      0500  OUTDIGIT:                                 ; OUTPUT ONE DIGIT
           52   D4    0500          CLRL    R2                        ; ZAP DIGIT SELECTOR
           03   11    0502          BRB     OUTCOM                    ; AND MERGE WITH COMMON
                      0504
                      0504  OUTLONG:                                  ; OUTPUT LONGWORD
```

```
           52    1C   D0 0504            MOVL     #28,R2                  ; SET DIGIT SELECTOR
                         0507   OUTCOM:                                   ; FORMAT IT
        54    08 AB   9E 0507            MOVAB    OUTBUF-B(R11),R4        ; GET ADDRESS OF OUTPUT BUFFER
 51   53   04    52   EF 050B   10$:     EXTZV    R2,#4,R3,R1             ; GET DIGIT
      84   FC89 CF41  90 0510            MOVB     PRIMARY[R1],(R4)+       ; BUFFER IT
           52    04   C2 0516            SUBL     #4,R2                   ; NEXT DIGIT
                 F0   18 0519            BGEQ     10$                     ; DO ALL REQUESTED
                 64   94 051B            CLRB     (R4)                    ; MARK END OF BUFFER
        54    08 AB   9E 051D   OUTZBUF:MOVAB     OUTBUF-B(R11),R4        ; GET START OF BUFFER
                         0521
                         0521   OUTZSTRING:                               ; OUTPUT ASCIZ STRING
                         0521            .IF      NDF,SW_PROCESS
           50    84   9A 0521            MOVZBL   (R4)+,R0                ; GET A CHAR
                 04   13 0524            BEQL     10$                     ; BR IF DONE
                 0C   10 0526            BSBB     OUTCHAR                 ; OUTPUT CHAR
                 F7   11 0528            BRB      OUTZSTRING              ; CONTINUE
                      05 052A   10$:     RSB                              ; RETURN IF DONE
                         052B            .IFF
                         052B            PUSHL    R5                      ; Save a register.
                         052B            LOCC     #0, #256, (R4)          ; Locate the terminating zero.
                         052B            SUBL3    R4, R1, R5              ; Compute the number of bytes to write.
                         052B            BEQL     90$                     ; Branch if zero bytes to write.
                         052B   50$:     $QIOW_S  EFN=#30,-               ; Write whole buffer.
                         052B                     CHAN=TTCHAN,-
                         052B                     FUNC=#IO$_WRITEVBLK,-
                         052B                     P1=(R4),-
                         052B                     P2=R5
                         052B            CMPW     R0,#SS$_INSFMEM         ; If any resource error occurs,
                         052B            BEQL     60$                     ; wait for an I/O completion
                         052B            CMPW     R0,#SS$_EXQUOTA         ; and try again.
                         052B            BEQL     60$
                         052B            CMPW     R0,#SS$_EXQUOTASTRT
                         052B            BLSSU    90$
                         052B            CMPW     R0,#SS$_EXQUOTAEND
                         052B            BGTRU    90$
                         052B   60$:     $WAITFR_S  EFN=#30
                         052B            BRB      50$
                         052B   90$:     POPL     R5                      ; Restore saved register.
                         052B            RSB
                         052B            .ENDC
                         052B
                         052B
                         052B   OUTBSLSH:                                 ; OUTPUT BACK SLASH
        50    5C 8F   9A 052B            MOVZBL   #BSLSH,R0               ; SET CHARACTER CODE
                 03   11 052F            BRB      OUTCHAR                 ; AND OUTPUT IT
           50    58   9A 0531   OUTR8:   MOVZBL   R8,R0                   ; GET CHAR TO OUTPUT
                         0534   OUTCHAR:                                  ; OUTPUT CHAR IN R0
                         0534            .IF      NDF,SW_PROCESS          ;
                 5C   D5 0534            TSTL     AP                      ; CHECK FOR CONSOLE
                 06   12 0536            BNEQ     10$                     ; NO, USE DEVICE DIRECTLY
        00000000'GF   17 0538            JMP      G^CON$PUTCHAR           ; OUTPUT TO THE CONSOLE TERMINAL
                         053E
        51    04 AC   B0 053E   10$:     MOVW     OUTCR(AP),R1            ; GET STATUS
        F8 51    07   E1 0542            BBC      #7,R1,10$               ; WAIT FOR READY
        06 AC    50   90 0546            MOVB     R0,OUTB(AP)             ; OUTPUT CHAR
                         054A            .IFF                             ; FALSE FOR PROCESS VERSION
                         054A            PUSHL    R0                      ; BUFFER CHARACTER ON STACK
```

```
               054A  50$:      MOVL    SP,R0                   ; SAVE POINTER TO IT
               054A            $QIOW_S EFN=#30,-               ;
               054A                    CHAN=TTCHAN,-           ;
               054A                    FUNC=#IO$_WRITEVBLK,-   ;
               054A                    P1=(R0),-               ; BUFFER ADDRESS
               054A                    P2=#1                   ; ONE CHARACTER
               054A            CMPW    R0,#SS$_INSFMEM         ; If any resource error occurs,
               054A            BEQL    60$                     ; wait for an I/O completion
               054A            CMPW    R0,#SS$_EXQUOTA         ; and try again.
               054A            BEQL    60$
               054A            CMPW    R0,#SS$_EXQUOTASTRT
               054A            BLSSU   90$
               054A            CMPW    R0,#SS$_EXQUOTAEND
               054A            BGTRU   90$
               054A  60$:      $WAITFR_S  EFN=#30
               054A            BRB     50$
               054A  90$:      POPR    #^M<R0>                 ; RESTORE CHARACTER
               054A            .ENDC                           ;
            05 054A            RSB                             ; AND RETURN
               054B  OUTSPACE:
   50  20  9A  054B            MOVZBL  #32,R0                  ; SET CODE FOR SPACE
       E4  11  054E            BRB     OUTCHAR                 ; AND SEND IT
   50  0D  9A  0550  CRLF:     MOVZBL  #CR,R0                  ; RETURN
       DF  10  0553            BSBB    OUTCHAR                 ; SEND IT
   50  0A  9A  0555            MOVZBL  #LF,R0                  ; LINE FEED
       DA  11  0558            BRB     OUTCHAR                 ; SEND IT
               055A
               055A
```

```
                    055A          .SBTTL  GETCHAR - GET INPUT CHARACTER ROUTINE
                    055A
                    055A     ;
                    055A     ;        GETCHAR - GET INPUT CHARACTER
                    055A     ;
                    055A     ; OUTPUT:
                    055A     ;        R8 - INPUT CHARACTER
                    055A     ;        R9 - BUFFER POINTER UPDATED (BUFFER IN ASCIZ FORMAT)
                    055A     ;
                    055A
                    055A GETCHAR:                                   ;
      58    89  9A  055A          MOVZBL  (R9)+,R8                  ; GET NEXT CHARACTER
            01  13  055D          BEQL    10$                       ; READ IF NONE AVAIL
            05      055F          RSB
59    84 AB     9E  0560 10$:     MOVAB   INBUF-B(R11),R9           ; SET ADDRESS OF INPUT BUFFER
                    0564          .IF     NDF,SW_PROCESS            ;
            5C  D5  0564 20$:     TSTL    AP                        ; CHECK FOR CONSOLE
            0B  12  0566          BNEQ    30$                       ; YES
  00000000'GF   16  0568          JSB     G^CON$GETCHAR             ; GET A CHARACTER FROM THE CONSOLE TERMINAL
      58    50  90  056E          MOVB    R0,R8                     ;
            0D  11  0571          BRB     60$                       ; CONTINUE IN COMMON
      50    6C  B0  0573 30$:     MOVW    RDCR(AP),R0               ; GET STATUS
   F9 50    07  E1  0576 40$:     BBC     #7,R0,30$                 ; WAIT FOR READY
   58    02 AC  90  057A          MOVB    RDBUF(AP),R8              ; GET CHARACTER
            00  11  057E          BRB     60$                       ; MERGE WITH COMMON
                    0580          .IFF                              ; FALSE IF PROCESS VERSION
                    0580 15$:     MOVAL   TTITMLST,R0               ; get the relocateable address
                    0580          $QIOW_S EFN=#31,-                 ;
                    0580                  CHAN=TTCHAN,-             ; INPUT DEVICE CHANNEL
                    0580                  IOSB=TTIOSB,-             ; IO STATUS BLOCK
                    0580                  FUNC=#<IO$_READVBLK!IO$M_EXTEND>,-     ;
                    0580                  P1=(R9),-                 ; BUFFER ADDRESS
                    0580                  P2=#80,-                  ; READ SIZE
                    0580                  P5=R0,-
                    0580                  P6=#TTITMLSTLEN
                    0580          CMPW    R0,#SS$_INSFMEM           ; If any resource error occurs,
                    0580          BEQL    760$                      ; wait for an I/O completion
                    0580          CMPW    R0,#SS$_EXQUOTA           ; and try again.
                    0580          BEQL    760$
                    0580          CMPW    R0,#SS$_EXQUOTASTRT
                    0580          BLSSU   790$
                    0580          CMPW    R0,#SS$_EXQUOTAEND
                    0580          BGTRU   790$
                    0580 760$:    $WAITFR_S  EFN=#31
                    0580          BRB     15$
                    0580 790$:
                    0580          MOVZWL  TTIOSB+2,R0               ; GET SIZE READ
                    0580          MOVB    TTIOSB+4,(R0)+[R9]        ; BUFFER TERMINATOR
                    0580          CLRB    (R9)[R0]                  ; MARK END OF BUFFER
                    0580          MOVL    R9,R2                     ; POINT TO START OF STRING
                    0580 20$:     MOVZBL  (R2)+,R8                  ; GET A CHARACTER
                    0580          BEQL    15$                       ; EMPTY, READ SOME MORE
                    0580          .ENDC                             ;
58    80 8F     8A  0580 60$:     BICB    #^X80,R8                  ; STRIP PARITY
7F 8F     58    91  0584          CMPB    R8,#RUBOUT                ; CHECK FOR RUBOUT
          15    12  0588          BNEQ    90$                       ; NO
03 6A     06    E2  058A          BBSS    #V_RUB,(R10),70$          ; SET START OF RUBOUT SEQUENCE
```

```
            FF9A  30  058E         BSBW    OUTBSLSH              ; OUTPUT BACK SLASH
         58   79  9A  0591  70$:   MOVZBL  -(R9),R8              ; GET RUBBED OUT CHAR
              04  12  0594         BNEQ    80$                   ; SKIP INC
              59  D6  0596         INCL    R9                    ; POINT AT START OF BUFFER
              CA  11  0598         BRB     20$                   ; AND GET ANOTHER
            FF94  30  059A  80$:   BSBW    OUTR8                 ; OUTPUT RUBBED OUT CHAR
              C5  11  059D         BRB     20$                   ; AND GET ANOTHER
      03 6A   06  E5  059F  90$:   BBCC    #V_RUB,(R10),100$     ; TERMINATE RUBOUT SEQUENCE
            FF85  30  05A3         BSBW    OUTBSLSH              ; OUTPUT BACK SLASH
      03 58   06  E1  05A6  100$:  BBC     #6,R8,110$            ; BR IF NOT ALPHA
         58   20  8A  05AA         BICB    #32,R8                ; SET TO UPPER CASE
                      05AD  110$:                                ;
                      05AD         .IF     NDF,SW_PROCESS        ;
            FF81  30  05AD         BSBW    OUTR8                 ; ECHO CHARACTER
                      05B0         .ENDC                         ;
         89   58  90  05B0         MOVB    R8,(R9)+              ; BUFFER NEW CHAR
FBFE CF  0A   58  3A  05B3         LOCC    R8,#NTERM,TERM        ; CHECK FOR TERMINATOR
              A9  13  05B9         BEQL    20$                   ; NOT A TERMINATOR
         58   0D  91  05BB         CMPB    #CR,R8                ; IS CHAR = RETURN
              03  12  05BE         BNEQ    120$                  ; NO,
            FF8D  30  05C0         BSBW    CRLF                  ; YES, SEND CR/LF
              69  94  05C3  120$:  CLRB    (R9)                  ; MARK END OF BUFFER
      59   84 AB  9E  05C5         MOVAB   INBUF-B(R11),R9       ; RESTORE BUFFER BASE
            FF8E  31  05C9         BRW     GETCHAR               ; AND TRY AGAIN
```

```
                       05CC           .SBTTL  PLUS/MINUS OPERATORS
                       05CC   ;
                       05CC   ;       PLUS/MINUS OPERATORS
                       05CC   ;
                       05CC   BLANK:                                  ; SAME AS PLUS
                       05CC   OPERATOR:                               ;
               FC8C 30 05CC           BSBW    ENDEXPR                 ; END EXPR
    FF AB   50   12 83 05CF           SUBB3   #OPERBAS,R0,OPER-B(R11) ; SET OPERATOR
                    05 05D4           RSB                             ; RETURN
                       05D5   ;
                       05D5   ;       MONADIC MINUS - NEGATE
                       05D5   ;
                 56 D5 05D5   NEGATE: TSTL    R6                      ; TEST ACCUMULATOR
                 03 13 05D7           BEQL    5$                      ; EMPTY
               FC7F 30 05D9           BSBW    ENDEXPR                 ; OTHERWISE PERFORM OPERATION
        6A    80 8F 8C 05DC   5$:     XORB    #<1@V_NEGATE>,(R10)     ; TOGGLE NEGATE FLAG
                    05 05E0   10$:    RSB                             ; AND RETURN
                       05E1
                       05E1
```

```
                              05E1          .SBTTL  TAB - INDIRECT DISPLAY
                              05E1    ;
                              05E1    ;       TAB
                              05E1    ;
                              05E1    ;
           6B   04 AB   D0    05E1    TAB:    MOVL    QUAN-B(R11),CURDOT-B(R11)       ; GO INDIRECT
 50   6A   01   0F      EF    05E5            EXTZV   #V_PRMODE,#1,(R10),R0   ; GET PROCESSOR REGISTER MODE
 6A   01   1F   50      F0    05EA            INSV    R0,#V_PREG,#1,(R10)     ; AND COPY TO SEMI-PERMANENT FLAG
                13      11    05EF            BRB     LOCP                    ; AND DISPLAY IT
                              05F1
                              05F1    ;
                              05F1    ;       ESCAPE - DISPLAY PREVIOUS LOCATION
                              05F1    ;
                              05F1
                              05F1    ESCAP:                                  ;
      0F   6A   0D      E0    05F1            BBS     #V_INSTR,(R10),LOCP     ; BRANCH IF INSTRUCTION MODE
           51   01      D0    05F5            MOVL    #1,R1                   ; ASSUME UNIT INCREMENT
                6A      D5    05F8            TSTL    (R10)                   ; CHECK FOR PROCESSOR REGISTER
                05      19    05FA            BLSS    10$                     ; YES, USE UNIT INCREMENT
      51   51   FE AB   9C    05FC            ROTL    CURTYPE-B(R11),R1,R1    ; FORM INCREMENT
           6B   51      C2    0601    10$:    SUBL    R1,CURDOT-B(R11)        ; AND SUBTRACT FROM DOT
              FD44      31    0604    LOCP:   BRW     LOCPROMPT               ; PROMPT WITH CONTENT
```

```
                    0607  1083          .SBTTL  DISPLAY INSTRUCTION RANGE
                    0607  1084  ;
                    0607  1085  ;       'I' - DISPLAY RANGE OF INSTRUCTIONS
                    0607  1086  ;
           FCDF 30  0607  1087  INSTR:  BSBW    ENDFIELD                    ; TERMINATE FIELD
        6A   02 8A  060A  1088          BICB    #1@V_ASCII,(R10)            ; CLEAR CHARACTER DISPLAY MODE
     06 6A   08 E0  060D  1089          BBS     #V_FT,(R10),5$              ; ADDRESS SPECIFIED?
     6B   04 AB D0  0611  1090          MOVL    QUAN-B(R11),CURDOT-B(R11) ; EXAMINE AT Q IF UNSPECIFIED
             04 11  0615  1091          BRB     10$
     6B   D8 AB D0  0617  1092  5$:     MOVL    F1-B(R11),CURDOT-B(R11)   ; IF ADDRESS SPECIFIED, SET NEW DOT
   6A   2000 8F A8  061B  1093  10$:    BISW    #1@V_INSTR,(R10)            ; SET INSTRUCTION DISPLAY MODE
           FD60 30  0620  1094          BSBW    OUTINS                      ; DISPLAY INSTRUCTION
     0B 6A   09 E1  0623  1095          BBC     #V_F2,(R10),30$             ; IF NO RANGE SPECIFIED, EXIT
     6B   DC AB D1  0627  1096  20$:    CMPL    F2-B(R11),CURDOT-B(R11)   ; END OF RANGE?
             05 15  062B  1097          BLEQ    30$                         ; BRANCH IF DONE
           FD19 30  062D  1098          BSBW    NEXTLOC                     ; OUTPUT NEXT INSTRUCTION
             F5 11  0630  1099          BRB     20$                         ; LOOP UNTIL DONE
             0F 11  0632  1100  30$:    BRB     RESET                       ; RESET SCANNER
```

```
                      0634  1102          .SBTTL  EQUALS - DISPLAY VALUE
                      0634  1103  ;
                      0634  1104  ;       EQUALS - VALUE DISPLAY
                      0634  1105  ;
                      0634  1106  EQUALS:                                  ;
                      0634  1107          .ENABL  LSB
            FCB2  30  0634  1108          BSBW    ENDFIELD                 ; TERMINATE FIELD
      05 6A   08  E1  0637  1109          BBC     #V_F1,(R10),10$          ; IGNORE IF FIELD BLANK
   04 AB   D8 AB  D0  063B  1110  EQL1:   MOVL    F1-B(R11),QUAN-B(R11)    ; SET QUANTITY
            FD14  30  0640  1111  10$:    BSBW    OUTPUT                   ; OUTPUT IT
                      0643  1112  ;       BRB     RESET                    ; RESET SCANNER
                      0643  1113          .DSABL  LSB                      ;
                      0643  1114
                      0643  1115  ;
                      0643  1116  ;       RESET
                      0643  1117  ;
                      0643  1118
6A   00FFDF80 8F  CA  0643  1119  RESET:  BICL    #^X0FFDF80,(R10)         ; CLEAR FIELD AND NEGATE FLAGS
           FC AB  94  064A  1120          CLRB    FCTR-B(R11)              ; CLEAR FIELD COUNTER
              56  7C  064D  1121          CLRQ    R6                       ; RESET ACCUMULATORS
              05  064F  1122          RSB                              ; RETURN
```

```
                          0650          .SBTTL  SEMI - SECONDARY COMMAND SET
                          0650  ;
                          0650  ;       SEMI
                          0650  ;
                          0650
                          0650  SECOND:                                     ;
                     58   0650          .ASCII  /X/                         ; X REGISTER SET/DISPLAY
                     50   0651          .ASCII  /P/                         ; P - PROCEED
                     4D   0652          .ASCII  /M/                         ; M - SET MODIFY FLAG
                     49   0653          .ASCII  /I/                         ; I - PROGRAM COUNTER
                     47   0654          .ASCII  /G/                         ; G - GO, START
                     45   0655          .ASCII  /E/                         ; E - EXECUTE STRING
                     42   0656          .ASCII  /B/                         ; B - SET/CLR BREAKPOINT
               00000007   0657  NSEC=.-SECOND                               ; NUMBER OF SECONDARY COMMANDS
                          0657
                          0657  SEMI:                                       ;
          6A    01   8A   0657          BICB    #<1@V_OPEN>,(R10)           ; CLEAR OPEN FLAG
              FC8C   30   065A          BSBW    ENDFIELD                    ; TERMINATE FIELD
              FEFA   30   065D          BSBW    GETCHAR                     ; GET SECONDARY COMMAND CHAR
 EB AF    07    58   3A   0660          LOCC    R8,#NSEC,SECOND             ; LOCATE SECONDARY COMMAND
                          0665  10$:    CASE    R0,LIMIT=#1,<-              ; SWITCH ON TYPE
                          0665                  BRKPOINT,-                  ; SET BREAKPOINT
                          0665                  EXECUTE,-                   ; EXECUTE STRING
                          0665                  GO,-                        ; SEMI-G, GO
                          0665                  PROGCTR,-                   ; SEMI-I, INSTRUCTION CONTER
                          0665                  MFYFLGS,-                   ; SEMI-M, MODIFY FLAG
                          0665                  PROCED,-                    ; SEMI-P, PROCEED
                          0665                  XSET,-                      ; SET XREGISTER
                          0665                  >                           ;
              FB5C   31   0677  ERR2:   BRW     ERROR                       ; ERROR
```

```
                          067A         .SBTTL  LEFT BRACKET - MODE SELECTION
                          067A     ;
                          067A     ;
                          067A     ;       LEFT BRACKET
                          067A     ;
                          067A     MODES:                                  ; MODE CHARACTER LIST
                      49  067A             .ASCII  /I/                     ; INSTRUCTION MODE, VARIABLE LENGTH
                      43  067B             .ASCII  /C/                     ; CHARACTER, CURRENT LENGTH
                      4C  067C             .ASCII  /L/                     ; LONG, HEX
                      57  067D             .ASCII  /W/                     ; WORD, HEX
                      42  067E             .ASCII  /B/                     ; BYTE, HEX
           00000005       067F     NMODES=.-MODES                          ; NUMBER OF MODE CHARACTERS
                          067F
                          067F
                          067F     LBRACKET:                               ; MODE SELECTION
               FED8   30  067F             BSBW    GETCHAR                 ; GET MODE CHAR
F3 AF     05     58   3A  0682             LOCC    R8,#NMODES,MODES        ; CONVERT TO INDEX
                 EE   13  0687             BEQL    ERR2                    ; NOT FOUND, ERROR
          04     50   D1  0689             CMPL    R0,#4                   ; CHECK FOR 'C'
                 10   13  068C             BEQL    10$                     ; BRANCH IF 'C'
                 17   14  068E             BGTR    20$                     ; BRANCH IF 'I'
FE AB     50     01   83  0690             SUBB3   #1,R0,CURTYPE-B(R11)    ; SET MODE
   6A   2000     8F   AA  0695             BICW    #1@V_INSTR,(R10)        ; CLEAR INSTRUCTION DISPLAY MODE
          6A     02   8A  069A     5$:     BICB    #<1@V_ASCII>,(R10)      ; CLEAR CHARACTER DISPLAY MODE
                      05  069D             RSB                             ; RETURN
          6A     02   88  069E     10$:    BISB    #<1@V_ASCII>,(R10)      ; SET CHARACTER DISPLAY MODE
   6A   2000     8F   AA  06A1             BICW    #1@V_INSTR,(R10)        ; CLEAR CHARACTER DISPLAY MODE
                      05  06A6             RSB
   6A   2000     8F   A8  06A7     20$:    BISW    #1@V_INSTR,(R10)        ; SET INSTRUCTION DISPLAY MODE
                 EC   11  06AC             BRB     5$
```

```
                              06AE  1186          .SBTTL  SINGLE STEP
                              06AE  1187  ;
                              06AE  1188  ;       STEP
                              06AE  1189  ;
   6A   02   03   01     F0   06AE  1190  STEP:   INSV    #1,#V_TBIT,#2,(R10)       ; CLR V_ATBRK, SET V_TBIT
   6A     80008000 8F    CA   06B3  1191          BICL    #<<1@V_PRMODE>!<1@V_PREG>>,(R10)       ; CLEAR PROCESSOR REGISTER M
                         04   06BA  1192          RET                               ; AND RETURN
```

```
                              06BB          .SBTTL  STEPOVER - STEP OVER ROUTINE CALL
                              06BB   ;
                              06BB   ;      STEPOVER
                              06BB   ;
                              06BB   STEPOVER:
        50    54 BB     9A    06BB          MOVZBL  @SAVPC-B(R11),R0                ; GET NEXT INSTRUCTION TO EXECUTE
  6A  80008000 8F       CA    06BF          BICL    #<<1@V_PRMODE>!<1@V_PREG>>,(R10)        ; CLEAR PROCESSOR REGISTER M
    51      FACF CF     9E    06C6          MOVAB   OVEROPCODES,R1                  ; ADDRESS OF LIST OF OPCODES
            52    05    9A    06CB          MOVZBL  #OVEROPCLEN,R2                  ; SIZE OF TABLE
            81    50    91    06CE   10$:   CMPB    R0,(R1)+                        ; MATCH?
                  05    13    06D1          BEQL    20$                             ; BRANCH IF FOUND
            F8    52    F5    06D3          SOBGTR  R2,10$                          ; LOOP UNTIL FOUND
                  D6    11    06D6          BRB     STEP                            ; IF NOT A ROUTINE CALL, NORMAL STEP
  52    00000000'GF     9E    06D8   20$:   MOVAB   G^LIB$INS_DECODE,R2             ; GET ADDRESS OF FOLLOWING INSTRUCTION
                  1E    13    06DF          BEQL    30$                             ; IF NOT AVAILABLE, ERROR
            54    AB    DD    06E1          PUSHL   SAVPC-B(R11)                    ; COPY ADDRESS OF INSTRUCTION STREAM
                  00    DD    06E4          PUSHL   #0                              ; PUSH NULL DESCRIPTOR
                  5E    DD    06E6          PUSHL   SP                              ; ADDRESS OF OUTPUT DESCRIPTOR
            08    AE    DF    06E8          PUSHAL  8(SP)                           ; ACCESS INSTRUCTION STREAM DIRECTLY
            62    02    FB    06EB          CALLS   #2,(R2)                         ; FIND ADDRESS OF FOLLOWING INSTRUCTION
            0B    50    E9    06EE          BLBC    R0,25$                          ; IF NOT INTERPRETABLE, ERROR
                              06F1          .IF     DF,SW_PROCESS                   ;
                              06F1          MOVL    4(SP),R4                        ; GET ADDRESS OF NEXT INSTRUCTION
                              06F1          MOVL    R4,R5                           ; MAKE END=START
                              06F1          BSBW    SETWRT                          ; MAKE INSTRUCTION WRITABLE
                              06F1          .ENDC
                  03    BA    06F1          POPR    #^M<R0,R1>                      ; GET UPDATED STREAM POINTER
            61    61    90    06F3          MOVB    (R1),(R1)                       ; ERROR IF UNABLE TO WRITE BREAKPOINT
       FA0D CF    51    D0    06F6          MOVL    R1,OVRADR                       ; SET TEMPORARY BREAKPOINT
                        04    06FB          RET                                     ; START EXECUTION
                              06FC
            5E    08    C0    06FC   25$:   ADDL    #8,SP                           ; CLEAN STACK
              FAD4      31    06FF   30$:   BRW     ERROR                           ; REPORT ERROR - UNABLE TO STEP OVER
```

```
                         0702              .SBTTL  BRKPOINT - SET/CLEAR BREAKPOINTS
                         0702        ;
                         0702        ;        BRKPOINT
                         0702        ;
                         0702        BRKPOINT:
       59 6A    08   E1  0702              BBC     #V_F1,(R10),SHOBRK           ; DISPLAY BREAKPOINTS
       13 6A    09   E0  0706              BBS     #V_F2,(R10),20$              ; YES, IT WAS SPECIFIED
          52    01   D0  070A              MOVL    #1,R2                        ; INIT INDEX
        F9D2 CF42    D5  070D  10$:        TSTL    BRKADR[R2]                   ; FIND FREE SLOT
                14   13  0712              BEQL    30$                          ; YES, GOT ONE
  FFF3 52   01  08   F1  0714              ACBL    #NBRK,#1,R2,10$              ; CHECK THEM ALL
              FAB9   31  071A  15$:        BRW     ERROR                        ; ERROR
       52    DC AB   D0  071D  20$:        MOVL    F2-B(R11),R2                 ; GET BRKPOINT NUMBER
                EA   13  0721              BEQL    10$                          ; NULL FIELD, SCAN FOR SLOT
          52    08   D1  0723              CMPL    #NBRK,R2                     ; CHECK FOR LEGAL
                F2   19  0726              BLSS    15$                          ; OUT OF RANGE
        F9E4 CF42    D4  0728  30$:        CLRL    BRKDSP[R2]                   ; CLEAR DISPLAY
        F9FF CF42    D4  072D              CLRL    BRKCOM[R2]                   ; CLEAR COMMAND ADDRESS
       50    D8 AB   D0  0732              MOVL    F1-B(R11),R0                 ; GET BREAKPOINT ADDRESS
                03   13  0736              BEQL    35$                          ; ALLOW CLEAR OF BREAKPOINT
                         0738              .IF     DF,SW_PROCESS                ;
                         0738              PUSHR   #^M<R0,R1,R2,R3,R4,R5>       ; SAVE REGISTERS FOR PROTECTION CHANGE
                         0738              MOVL    R0,R4                        ; SET START ADDRESS
                         0738              MOVL    R0,R5                        ; AND END ADDRESS
                         0738              BSBW    SETWRT                       ; SET PAGE WRITABLE
                         0738              MOVL    (SP),R0                      ; RESTORE BPT ADDRESS
                         0738              .ENDC
          60    60   90  0738              MOVB    (R0),(R0)                    ; TEST WRITABILITY OF ADDRESS
                         073B              .IF     DF,SW_PROCESS                ;
                         073B              BSBW    REPROT                       ; RESTORE PROTECTION
                         073B              POPR    #^M<R0,R1,R2,R3,R4,R5>       ; AND REGISTERS
                         073B              .ENDC
       0C 6A    0A   E1  073B  35$:        BBC     #V_F3,(R10),40$              ; DISPLAY SPECIFIED?
  F9CB CF42  E0 AB   D0  073F              MOVL    F3-B(R11),BRKDSP[R2]         ; SET DISPLAY START
                03   13  0746              BEQL    40$                          ; SKIP TEST IF NULL
             E0 BB   D5  0748              TSTL    @F3-B(R11)                   ; CHECK READABILITY
       07 6A    0B   E1  074B  40$:        BBC     #V_F4,(R10),45$              ; SKIP IF NO COMMAND ADDRESS
  F9DB CF42  E4 AB   D0  074F              MOVL    F4-B(R11),BRKCOM[R2]         ; SET COMMAND STRING
   F988 CF42    50   D0  0756  45$:        MOVL    R0,BRKADR[R2]                ; SAVE BREAKPOINT ADDRESS
              FEE4   31  075C              BRW     RESET                        ; RESET SCANNER AND RETURN
                         075F        ;
                         075F        ;        SHOBRK
                         075F        ;
                         075F        SHOBRK:                                    ;
          55    01   D0  075F              MOVL    #1,R5                        ; INIT INDEX FOR LOOP
    58   F97D CF45   D0  0762  10$:        MOVL    BRKADR[R5],R8                ; GET BREAKPOINT ADDRESS
                2E   13  0768              BEQL    20$                          ; SKIP IF NULL
          53    55   D0  076A              MOVL    R5,R3                        ; BREAKPOINT NUMBER
              FDE0   30  076D              BSBW    CRLF                         ; NEW LINE
              FD8D   30  0770              BSBW    OUTDIGIT                     ; BPT NUMBER
              FDD5   30  0773              BSBW    OUTSPACE                     ; SPACE
          53    58   D0  0776              MOVL    R8,R3                        ; ADDRESS OF BPT
              FD88   30  0779              BSBW    OUTLONG                      ; OUTPUT ADDRESS
              FDCC   30  077C              BSBW    OUTSPACE                     ; SPACE OVER
    53   F98D CF45   D0  077F              MOVL    BRKDSP[R5],R3                ; GET DISPLAY START
                03   13  0785              BEQL    15$                          ; NONE
              FD7A   30  0787              BSBW    OUTLONG                      ; OUTPUT DISPLAY START
```

```
        53   F9A2 CF45   D0  078A  1285 15$:   MOVL    BRKCOM[R5],R3          ; GET COMMAND STRING ADDRESS
                    06   13  0790  1286        BEQL    20$                    ; NONE
                  FDB6   30  0792  1287        BSBW    OUTSPACE               ; SPACE ANOTHER
                  FD6C   30  0795  1288        BSBW    OUTLONG                ; AND OUTPUT A LONGWORD
   FFC4 55     01   08   F1  0798  1289 20$:   ACBL    #NBRK,#1,R5,10$        ; DO THEM ALL
                  FDAF   31  079E  1290        BRW     CRLF                   ; AND EXIT THROUGH CRLF
```

```
                           07A1          .SBTTL  GO - START EXECUTION AT SPECIFIED LOCATION
                           07A1      ;
                           07A1      ;       GO
                           07A1      ;
      05 6A    08     E1   07A1 GO:      BBC     #V_F1,(R10),PROCED      ; JUST PROCEED IF NO VALUE
   54 AB    D8 AB     D0   07A5          MOVL    F1-B(R11),SAVPC-B(R11)  ; SET NEW PC
                           07AA      ;   BRW     PROCED                  ; FALL INTO PROCEED
                           07AA      ;
                           07AA      ;       PROCEED
                           07AA      ;
                           07AA PROCED:                                  ;
 6A    80008000 8F    CA   07AA          BICL    #<<1@V_PRMODE>!<1@V_PREG>>,(R10)        ; CLEAR PROCESSOR REGISTER M
                      04   07B1          RET                             ; RETURN
```

```
                         07B2          .SBTTL  SEMI-I, PC VALUE
                         07B2  ;
                         07B2  ;       SEMI-I
                         07B2  ;
            FAA6   30    07B2  COLON:  BSBW    ENDEXPR                 ; TERMINATE EXPRESSION
     F0 AB    57   D0    07B5          MOVL    R7,PID-B(R11)           ; SET PID FOR PROCESS
              56   7C    07B9          CLRQ    R6                      ; RESET ACCUMULATORS
              05         07BB          RSB                             ;
                         07BC
     51    EC AB   DE    07BC  MFYFLGS:MOVAL   MFYFLG-B(R11),R1        ; SET MODIFY FLAG ADDRESS
              17   11    07C0          BRB     VALUE                   ; SET/GET VALUE
          51  6B   DE    07C2  DOT:    MOVAL   CURDOT-B(R11),R1        ; SET ADDRESS OF DOT
     18 6A    1F   E1    07C5          BBC     #V_PREG,(R10),VALR      ; WAS IT PROCESSOR REGISTER?
     14 6A    0F   E2    07C9          BBSS    #V_PRMODE,(R10),VALR    ; YES, SET PROCESSOR REGISTER MODE
              12   11    07CD          BRB     VALR                    ; READ VALUE
     51    04 AB   DE    07CF  QUANT:  MOVAL   QUAN-B(R11),R1          ; SET QUANTITY ADDRESS
              0C   11    07D3          BRB     VALR                    ; READ VALUE
                         07D5  PROGCTR:                                ;
     51    54 AB   DE    07D5          MOVAL   SAVPC-B(R11),R1         ; SET PC ADDRESS
     04 6A    08   E1    07D9  VALUE:  BBC     #V_F1,(R10),VALR        ; SKIP IF NO VALUE
     61    D8 AB   D0    07DD          MOVL    F1-B(R11),(R1)          ; SET NEW VALUE FOR PC
          56  61   D0    07E1  VALR:   MOVL    (R1),R6 ; AND GET VALUE
            FA5B   31    07E4  VALI:   BRW     INFLD                   ; SET FIELD IN PROGRESS
                         07E7  REGISTER:
     55    18 AB   DE    07E7          MOVAL   SAVREG-B(R11),R5        ; SET BASE OF REGISTER AREA
              02   10    07EB          BSBB    REGCOM                  ; FETCH ADDRESS
              F5   11    07ED          BRB     VALI                    ; AND USE IT
            FD68   30    07EF  REGCOM: BSBW    GETCHAR                 ; GET SECOND CHAR
 F9A6 CF   10   58   3A  07F2          LOCC    R8,#16,PRIMARY          ; TRANSLATE TO HEX
                         07F8          .IF     DF,SW_PROCESS           ; FOR PROCESS VERSION
                         07F8          BNEQ    10$                     ; LEGAL HEX DIGIT
                         07F8          CMPW    #^A/XI/,-2(R9)          ; CHECK FOR EXIT COMMAND
                         07F8          BNEQ    ERR3                    ; NO, ERROR
                         07F8          TSTL    ASTEN-B(R11)            ; WERE ASTS ENABLED ON DELTA ENTRY?
                         07F8          BEQL    5$                      ; IF EQL NO, DON'T REENABLE THEM HERE
                         07F8          $SETAST_S       #1              ; YES, UNDO AST DISABLE BY DELTA
                         07F8  5$:     $EXIT_S EXITCODE                ; EXIT
                         07F8          .IFF
              6E   13    07F8          BEQL    ERR3                    ; ERROR, NOT HEX
                         07FA          .ENDC
                         07FA  10$:
     50    10 50   C3    07FA          SUBL3   R0,#16,R0               ; INVERT
       56   6540   DE    07FE          MOVAL   (R5)[R0],R6             ; ACCUMULATE
              05         0802          RSB                             ; RETURN
                         0803
     61 6A    09   E1    0803  XSET:   BBC     #V_F2,(R10),ERR3        ; ERROR IF NOT TWO FIELDS
 51  DC AB  04  00  EF   0807          EXTZV   #0,#4,F2-B(R11),R1      ; GET REGISTER NUMBER
   51  F943 CF41   DE    080D          MOVAL   XREGV[R1],R1            ; AND COMPUTE REGISTER ADDRESS
              C4   11    0813          BRB     VALUE                   ; PROCESS VALUE
                         0815  XREG:                                   ; X-REGISTER VALUE
     55  F93C CF   DE    0815          MOVAL   XREGV,R5                ; SET ADDRESS OF REGISTER VECTOR
              D3   10    081A          BSBB    REGCOM                  ; ADDRESS TO R6
          56  66   D0    081C          MOVL    (R6),R6                 ; GET VALUE
              C3   11    081F          BRB     VALI                    ; AND NOTE INPUT IN FIELD
                         0821          .ALIGN  LONG                    ; LONGWORD ALIGN EXCEPTION ROUTINES
                         0824  XDELACV:                                ; ACCESS VIOLATION HANDLER
                         0824  MCHK:                                   ; MACHINE CHECK
```

```
                            0824        .IF     NDF,SW_PROCESS          ;
                 5C    D5   0824        TSTL    AP                      ; CHECK FOR SIMULATOR
                 40    12   0826        BNEQ    ERR3                    ; YES, SKIP RESET
                            0828                                        ;
                            0828                                        ;
                            0828        CPUDISP <<780,CLR_780>,-        ; *DISPATCH ON CPU TYPE*
                            0828                 <750,CLR_750>,-        ;
                            0828                 <730,CLR_730>,-
                            0828                 <790,CLR_790>,-
                            0828                 <UV1,CLR_UV1>>,-
                            0828                 ENVIRON=XDELTA;
                            083F                                        ;
                            083F CLR_780:                               ; FOR 11/780:
           7E    30    DB   083F        MFPR    #PR$_SBIFS,-(SP)        ; GET ERROR BITS
        00 6E    19    E5   0842        BBCC    #25,(SP),10$            ; CLEAR ERROR 1ST PASS BIT
           30    8E    DA   0846 10$:   MTPR    (SP)+,#PR$_SBIFS        ; CLEAR SBI FAULT
                 1D    11   0849        BRB     CLR_END                 ; ERROR CLEARED
                            084B                                        ;
                            084B CLR_UV1:                               ; FOR MicroVAX I:
                            084B CLR_730:                               ; FOR 11/730:
                            084B CLR_750:                               ; FOR 11/750:
           26    0F    DA   084B        MTPR    #^XF,#PR$_MCESR         ; SET 1 TO CLEAR MCHECK ERROR SUMMARY
                 18    11   084E        BRB     CLR_END                 ;
                            0850 CLR_790:
     7E   0000004A 8F  DB   0850        MFPR    #PR$_EHSR,-(SP)         ; GET ERR HANDLING STATUS REGISTER
        00 6E    06    E5   0857        BBCC    #6,(SP),10$             ; CLEAR VMS ENTERED BIT
   0000004A 8F   8E    DA   085B 10$:   MTPR    (SP)+,#PR$_EHSR         ; WRITE BACK TO CLEAR
     00000000'EF       16   0862        JSB     SYSL$CLRSBIA            ; CLEAR SBIA ERROR BITS
                            0868                                        ;
                            0868 CLR_END:                               ; *END OF CPU-DEPENDENT CODE*
                            0868                                        ;
                            0868                                        ;
                            0868        .ENDC                           ;
                            0868 10$:                                   ;
               F96B    31   0868 ERR3:  BRW     ERROR                   ; AND DECLARE ERROR
                            086B
```

```
                         086B        .SBTTL  REGISTER SAVE AND RESTORE
                         086B
                         086B     ;
                         086B     ;       SAVE - SAVE TARGET REGISTERS, PC, PSL
                         086B     ;
                         086B     SAVE:                                          ;
                         086B        .IF     NDF,SW_PROCESS                      ;
                         086B        SETIPL  #31                                 ; DISABLE
    00000000'EF      16  086E        JSB     INI$WRITABLE                        ; MAKE THE SYSTEM WRITABLE
 F81F CF      50     7D  0874        MOVQ    R0,SAVREG                           ; SAVE R0,R1
   51    F823 CF     9E  0879        MOVAB   SAVR2,R1                            ; SETUP BASE FOR REMAINING REGS
                         087E        .IFF                                        ; FALSE IF PROCESS VERSION
                         087E        $SETAST_S       #0                          ; DISABLE ASTS
                         087E        PUSHAB  -(R0)                               ; SAVE ENABLE VALUE-1
                         087E        MOVPSL  R1                                  ; GET CURRENT PSL
                         087E        EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1       ; ISOLATE CURRENT MODE
                         087E        MULW    #CONTEXTSZ,R1                       ; COMPUTE OFFSET TO PROPER CONTEXT AREA
                         087E        MOVAB   SAVREG[R1],R1                       ; FORM ADDRESS OF REGISTER SAVE
                         087E        MOVL    8(AP),R0                            ; GET POINTER TO MECHANISM
                         087E        MOVQ    12(R0),(R1)+                        ; SAVE R0,R1
                         087E        .ENDC                                       ;
         81   52     7D  087E        MOVQ    R2,(R1)+                            ; SAVE R2,R3
         81   54     7D  0881        MOVQ    R4,(R1)+                            ; SAVE R4,R5
         81   56     7D  0884        MOVQ    R6,(R1)+                            ; SAVE R6,R7
         81   58     7D  0887        MOVQ    R8,(R1)+                            ; SAVE R8,R9
         81   5A     7D  088A        MOVQ    R10,(R1)+                           ; SAVE R10,R11
                         088D        .IF     NDF,SW_PROCESS                      ;
         81   5C     7D  088D        MOVQ    AP,(R1)+                            ; SAVE AP,FP
   81      0C AE     9E  0890        MOVAB   12(SP),(R1)+                        ; ASSUME KERNEL STACK
   81      04 AE     7D  0894        MOVQ    4(SP),(R1)+                         ; SAVE PC,PSL
                         0898        .IFF                                        ;
                         0898        MOVQ    8(FP),(R1)+                         ; SAVE AP,FP
                         0898        SUBL3   #1,@4(AP),R0                        ; GET NUMBER OF ARGS IN SIGNAL
                         0898        MOVAL   @4(AP)[R0],R0                       ; POINT TO PC,PSL
                         0898        MOVAL   8(R0),(R1)+                         ; COMPUTE SP
                         0898        MOVQ    (R0),(R1)+                          ; SAVE PC,PSL
                         0898        .ENDC                                       ;
                         0898        .IF     NDF,SW_PROCESS                      ;
         52   51     D0  0898        MOVL    R1,R2                               ; SAVE R1
    00000000'GF      16  089B        JSB     G^CON$OWNCTY                        ; ALLOCATE THE CONSOLE TERMINAL
         82   50     D0  08A1        MOVL    R0,(R2)+                            ; SAVE CONSOLE TRANSMIT STATUS
         82   51     D0  08A4        MOVL    R1,(R2)+                            ; SAVE CONSOLE RECVR STATUS
         51   52     D0  08A7        MOVL    R2,R1                               ; RESTORE R1
              5C     D4  08AA        CLRL    AP                                  ; ZAP DEVICE ADDRESS BASE
   5B    F7D0 CF     9E  08AC 20$:   MOVAB   B,R11                               ; AND DATA BASE ADDRESS
                         08B1        .IFF                                        ; FALSE FOR PROCESS VERSION
                         08B1        MOVAB   W^<B-<SAVPSL+4>>(R1),R11            ; SET BASE OF CONTEXT AREA
                         08B1        MOVL    (SP)+,ASTEN-B(R11)                  ; SAVE AST ENABLE
                         08B1        .ENDC                                       ;
   5A      D4 AB     9E  08B1        MOVAB   STATUS-B(R11),R10                   ; SET STATUS BASE
   59      84 AB     9E  08B5        MOVAB   INBUF-B(R11),R9                     ; POINT TO INPUT BUFFER
              69     94  08B9        CLRB    (R9)                                ; MAKE BUFFER EMPTY
                         08BB        .IF     NDF,SW_PROCESS                      ;
            0094     30  08BB        BSBW    GETSCB                              ; GET BASE OF SCB
 F8D1 CF   04 A0     D0  08BE        MOVL    4(R0),MCHKSAV                       ; SAVE ORIGINAL MCHK VECTOR
 04 A0    FF5C CF    9E  08C4        MOVAB   MCHK,4(R0)                          ; SET TO XDELTA VECTOR
 20 A0    FF56 CF    9E  08CA        MOVAB   XDELACV,^X20(R0)                    ; SET ACCESS VIOLATION VECTOR
```

```
       24 A0    FF50 CF    9E  08D0         MOVAB   XDELACV,^X24(R0)          ; SET PG FAULT VECTOR
       18 A0    FF4A CF    9E  08D6         MOVAB   XDELACV,^X18(R0)          ; SET RESERVED OPERAND HANDLER
50     08 AE      02       18  EF  08DC     EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,8(SP),R0     ; GET MODE
                  07       13  08E2         BEQL    30$                       ; CORRECT ALREADY IF KERNEL
             50   00       C0  08E4         ADDL    #PR$_KSP,R0               ; COMPUTE PROCESSOR REGISTER
          50 AB   50       DB  08E7         MFPR    R0,SAVSP-B(R11)           ; AND SAVE CORRECT SP
                               08EB         .ENDC                             ;
                FD55       31  08EB 30$:    BRW     RESET                     ; RESET SCANNER
                               08EE
                               08EE ;
                               08EE ;       RESTORE - RESTORE TARGET REGISTERS
                               08EE ;
                               08EE RESTORE:                                  ; RESTORE EVERYTHING
                               08EE         .IF     NDF,SW_PROCESS            ;
       04 AE      54 AB    7D  08EE         MOVQ    SAVPC-B(R11),4(SP)        ; SET PC,PSL
                               08F3         .IFF                              ; FALSE IF PROCESS
                               08F3         SUBL3   #1,@4(AP),R0              ; GET SIGNAL ARG COUNT
                               08F3         MOVAL   @4(AP)[R0],R0             ; COMPUTE ADDRESS OF PC,PSL
                               08F3         MOVQ    SAVPC-B(R11),(R0)         ; STORE UPDATED PC,PSL
                               08F3         .ENDC                             ;
                               08F3 RESTORR:                                  ; RESTORE REGISTERS ONLY
                               08F3         .IF     NDF,SW_PROCESS            ;
                  5D       10  08F3         BSBB    GETSCB                    ; GET BASE OF SCB
   20 A0    00000000'EF    9E  08F5         MOVAB   EXE$ACVIOLAT,^X20(R0)     ; RESTORE ACCESS VECTOR
   24 A0    00000000'EF    9E  08FD         MOVAB   MMG$PAGEFAULT,^X24(R0)    ; AND PAGE FAULT VECTOR
       04 A0    F88C CF    D0  0905         MOVL    MCHKSAV,4(R0)             ; RESTORE MACHINE CHECK VECTOR
   18 A0    00000000'EF    9E  090B         MOVAB   EXE$ROPRAND,^X18(R0)      ; RESTORE RESERVED OPERAND VECTOR
                  5C       B5  0913         TSTW    AP                        ; CHECK FOR CONSOLE
                  10       12  0915         BNEQ    10$                       ; NO, OTHER DEVICE
          50    5C AB      D0  0917         MOVL    SAVOCR-B(R11),R0          ; RESTORE INITIAL TX STATUS
          51    60 AB      D0  091B         MOVL    SAVRXCS-B(R11),R1         ; AND INITIAL RECEIVER STATE
          00000000'GF      16  091F         JSB     G^CON$RELEASECTY          ;
                  09       11  0925         BRB     20$                       ; MERGE WITH COMMON CODE
                               0927
       04 AC      5C AB    B0  0927 10$:    MOVW    SAVOCR-B(R11),OUTCR(AP)   ; RESTORE OUTPUT CSR
          6C      5E AB    B0  092C         MOVW    SAVRCR-B(R11),RDCR(AP)    ; AND INPUT CSR CONTENT
                               0930         .IFF                              ;
                               0930         PUSHL   ASTEN-B(R11)              ; SAVE AST ENABLE
                               0930         .ENDC                             ;
          51      20 AB    9E  0930 20$:    MOVAB   SAVR2-B(R11),R1           ; SET BASE FOR RESTORE
             52   81       7D  0934         MOVQ    (R1)+,R2                  ; RESTORE R2,R3
             54   81       7D  0937         MOVQ    (R1)+,R4                  ; RESTORE R4,R5
             56   81       7D  093A         MOVQ    (R1)+,R6                  ; RESTORE R6,R7
             58   81       7D  093D         MOVQ    (R1)+,R8                  ; RESTORE R8,R9
             5A   81       7D  0940         MOVQ    (R1)+,R10                 ; RESTORE R10,R11
                               0943         .IF     NDF,SW_PROCESS            ;
             5C   81       7D  0943         MOVQ    (R1)+,AP                  ; RESTORE AP,FP
       50    F74E CF       7D  0946         MOVQ    SAVREG,R0                 ; RESTORE R0,R1
                               094B         .IFF                              ; FALSE IF PROCESS VERSION
                               094B         MOVQ    (R1)+,8(FP)               ; SET NEW VALUES FOR AP,FP
                               094B         MOVL    8(AP),R0                  ; GET MECHANISM POINTER
                               094B         MOVQ    <SAVREG-SAVSP>(R1),12(R0)       ; STORE UPDATED R0,R1
                               094B         MOVPSL  R1                        ; GET CURRENT PSL
                               094B         EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1      ; GET CURRENT MODE
                               094B         BBCC    R1,DBGACTIVE,30$          ; CLEAR ACTIVE BIT FOR MODE
                               094B 30$:
                               094B         TSTL    (SP)+                     ; CHECK FOR AST ENABLE
```

```
                    094B          BEQL    35$                     ; NO
                    094B          $SETAST_S       #1              ; RE- ENABLE AST RECOGNITION
                    094B     35$:
                    094B          .ENDC                           ;
                    094B          .IF     NDF,SW_PROCESS          ;
00000000'EF     16  094B          JSB     INI$RDONLY              ; REPROTECT THE SYSTEM CODE
                    0951          .ENDC                           ;
                05  0951          RSB                             ; AND RETURN
```

```
                        0952        .SBTTL  GET SCB ADDRESS
                        0952
                        0952  ;
                        0952  ; SUBROUTINE GETSCB IS CALLED TO GET THE PHYSICAL OR VIRTUAL
                        0952  ; ADDRESS OF THE CURRENT SCB.
                        0952  ;
                        0952  ; INPUTS:         NONE
                        0952  ;
                        0952  ; OUTPUTS:        RO = SCB ADDRESS
                        0952  ;                 OTHER REGISTERS PRESERVED
                        0952  ;
                        0952
                        0952          .IF     NDF,SW_PROCESS          ; NOT FOR PROCESS VERSION
       50   38     DB   0952  GETSCB: MFPR    #PR$_MAPEN,RO           ; GET MAPPING STATUS
            05     12   0955          BNEQ    10$                     ; BRANCH IF MAPPING ENABLED
       50   11     DB   0957          MFPR    #PR$_SCBB,RO            ; ELSE GET PHY ADDR OF SCB
            07     11   095A          BRB     20$                     ; JOIN COMMON RETURN
  50   00000000'EF D0   095C  10$:    MOVL    EXE$GL_SCB,RO           ; IF MAPPING ENABLED, GET SCB VA
                   05   0963  20$:    RSB                             ; RETURN
                        0964          .ENDC                           ;
```

```
                              0964         .SBTTL  BPT TRAP HANDLER
                              0964  ;
                              0964  ;      HANDLE BREAKPOINT TRAPS
                              0964  ;
00 20 54 41 20 4B 52 42 20    0964  BMSG:  .ASCIZ  / BRK AT /               ; BREAK POINT MESSAGE
                              096D         .ALIGN  LONG                     ; LONGWORD ALIGNMENT
                              0970         .IF     NDF,SW_PROCESS           ; EXEC VERSION
                              0970  XDELBPT::                               ; XDELTA BPT ENTRY
                              0970         .IFF
                              0970  XDELBPT:                                ; DELTA BPT ENTRY
                              0970         .ENDC                            ;
              FEF8    30      0970         BSBW    SAVE                     ; SAVE REGS AND DISABLE
              00ED    30      0973         BSBW    GETBPTX                  ; GET INDEX OF BPT
                53    D5      0976         TSTL    R3                       ; CHECK FOR MATCH
                21    12      0978         BNEQ    10$                      ; YES, FOUND IT
   F788 CF   54 AB    D1      097A         CMPL    SAVPC-B(R11),OVRADR      ; IS THIS A TEMPORARY BREAKPOINT?
                10    13      0980         BEQL    20$                      ; BRANCH IF SO
              FF6E    30      0982         BSBW    RESTORR                  ; RESTORE REGISTERS ONLY
                              0985         .IF     NDF,SW_PROCESS           ;
        7E   06 AE    9A      0985         MOVZBL  6(SP),-(SP)              ; GET IPL
                              0989         ENBINT                           ; ENABLE
       00000000'EF    17      098C         JMP     EXE$BREAK                ; AND HANDLE NORMALLY
                              0992         .IFF                             ; FALSE IF PROCESS VERSION
                              0992  ;
                              0992  ;      ***** UNEXPECTED BREAKPOINT *****
                              0992         CLRL    R0                       ; RETURN FALSE
                              0992         RET                              ;
                              0992         .ENDC                            ;
                              0992  ;
                              0992  ; WE JUST HIT A TEMPORARY BREAKPOINT SET FROM A STEP-OVER
                              0992  ;
              0096    30      0992  20$:   BSBW    UNBRK                    ; RESTORE OPCODES, INCLUDING TEMP BRKPT
           F76F CF    D4      0995         CLRL    OVRADR                   ; REMOVE TEMPORARY BREAKPOINT
                40    11      0999         BRB     OUTPC                    ; AND PRETEND WE JUST STEPPED
                              099B
           6A   18    88      099B  10$:   BISB    #<<1@V_TBIT>!<1@V_ATBRK>>,(R10) ; SET STATUS
                              099E  30$:
              008A    30      099E         BSBW    UNBRK                    ; RESTORE OPCODES
    4D 58 AB    04    E0      09A1         BBS     #PSL$V_TBIT,SAVPSL-B(R11),PROCEED     ; PROCEED IF BPT AND TBIT
           55   53    D0      09A6         MOVL    R3,R5                    ; SAVE BPT NUMBER
              FBA4    30      09A9         BSBW    CRLF                     ; OUTPUT CR/LF PAIR
              FB51    30      09AC         BSBW    OUTDIGIT                 ; OUTPUT BPT NUMBER
        54   B2 AF    9E      09AF         MOVAB   BMSG,R4                  ; MSG ADDRESS
              FB6B    30      09B3         BSBW    OUTZSTRING               ; OUTPUT ASCIIZ
        53   54 AB    D0      09B6         MOVL    SAVPC-B(R11),R3          ; OUTPUT PC
              FB47    30      09BA         BSBW    OUTLONG                  ; OUTPUT HEX LONGWORD
   51   F74F CF45     D0      09BD         MOVL    BRKDSP[R5],R1            ; GET ADDRESS TO DISPLAY
                0B    13      09C3         BEQL    40$                      ; NONE
           6B   51    D0      09C5         MOVL    R1,CURDOT-B(R11)         ; SET AS CURRENT DOT
      6A   2000 8F    AA      09C8         BICW    #1@V_INSTR,(R10)         ; CLEAR INSTRUCTION DISPLAY MODE
              F97B    30      09CD         BSBW    LOCPROMPT                ; AND DISPLAY
   51   F75C CF45     D0      09D0  40$:   MOVL    BRKCOM[R5],R1            ; GET COMMAND STRING ADDRESS
                03    13      09D6         BEQL    OUTPC                    ; NONE OUTPUT INSTRUCTION AT PC
           59   51    D0      09D8         MOVL    R1,R9                    ; SET TO SCAN STORED COMMAND
                              09DB  OUTPC:                                  ; OUTPUT PC INSTRUCTION & GET COMMANDS
        6B   54 AB    D0      09DB         MOVL    SAVPC-B(R11),CURDOT-B(R11)     ; SET ADDRESS
                              09DF         IFNORD  #4,@CURDOT-B(R11),GETCMD ; SKIP DISPLAY IF NOT READABLE
```

```
 6A  2000 8F   A8  09E6             BISW    #1@V_INSTR,(R10)            ; SET TO INSTRUCTION DISPLAY MODE
          F95D  30  09EB             BSBW    LOCPROMPT                   ; PROMPT WITH ADDRESS/INSTRUCTION
                    09EE   GETCMD:                                       ; GET COMMANDS
 F7DF CF    6C  FA  09EE             CALLG   (AP),DCOM                   ; PERFORM DEBUG COMMANDS
                    09F3   PROCEED:                                      ; PROCEED
            4B  10  09F3             BSBB    SETBRK                      ; SET BREAKPOINTS
    09 6A   03  E5  09F5             BBCC    #V_TBIT,(R10),50$           ; TEST AND CLR TRACE FLAG
 00 58 AB   04  E2  09F9   30$:      BBSS    #PSL$V_TBIT,SAVPSL-B(R11),40$    ; SET TBIT
                    09FE   40$:
                    09FE             .IF     DF,SW_PROCESS               ; FOR PROCESS VERSION
                    09FE             CMPB    #2,@SAVPC-B(R11)            ; CHECK FOR REI OPCODE
                    09FE             BNEQ    45$                         ; NO, NOTHING SPECIAL
                    09FE             EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,SAVPSL-B(R11),R0     ; GET NEW MODE
                    09FE             MULW    #CONTEXTSZ,R0               ; SCALE BY PER MODE CONTEXT AREA SIZE
                    09FE             MOVAB   STATUS[R0],R10  ; POINT TO NEW FLAGS
                    09FE             .ENDC                               ;
    00 6A   05  E2  09FE   45$:      BBSS    #V_TBITOK,(R10),50$         ; SET TBIT EXPECTED
          FEE9  30  0A02   50$:      BSBW    RESTORE                     ; RESTORE EVERYTHING
                    0A05             .IF     NDF,SW_PROCESS              ;
                02  0A05             REI                                 ; AND RETURN
                    0A06             .IFF                                ; FALSE IF PROCESS VERSION
                    0A06             MOVL    #1,R0                       ; RETURN TRUE
                    0A06             RET                                 ;
                    0A06             .ENDC                               ;
                    0A06
```

```
                        0A06                .SBTTL  TBIT EXCEPTION HANDLER
                        0A06        ;
                        0A06        ;       HANDLER FOR TBIT EXCEPTION
                        0A06        ;
                        0A06                .ALIGN  LONG                            ; LONGWORD ALIGNED
                        0A08                .IF     NDF,SW_PROCESS                  ;
                        0A08        XDELTBIT::                                      ; XDELTA TBIT HANDLER
                        0A08                .IFF                                    ;
                        0A08        XDELTBIT:                                       ;
                        0A08                .ENDC                                   ;
           FE60    30   0A08                BSBW    SAVE                            ; SAVE AND DISABLE
     10 6A   05    E4   0A0B                BBSC    #V_TBITOK,(R10),XDELDBG         ; BR IF TBIT EXPECTED
           FEE1    30   0A0F                BSBW    RESTORR                         ; RESTORE REGISTERS
                        0A12                .IF     NDF,SW_PROCESS                  ;
     7E   06 AE    9A   0A12                MOVZBL  6(SP),-(SP)                     ; GET IPL FOR ENABLE
                        0A16                ENBINT                                  ; ENABLE
      00000000'EF  17   0A19                JMP     EXE$TBIT                        ; OTHERWISE LET EXEC HANDLE
                        0A1F                .IFF                                    ; FALSE IF PROCESS VERSION
                        0A1F                CLRL    R0                              ; RESIGNAL
                        0A1F                RET                                     ; UNEXPECTED TBIT EXCEPTION
                        0A1F                .ENDC                                   ;
                        0A1F        XDELDBG:                                        ; COMMON WITH DEBUG EXCEPTION
     58 AB   10    CA   0A1F                BICL    #<1@PSL$V_TBIT>,SAVPSL-B(R11)   ; CLEAR TBIT IN PSL
             06    10   0A23                BSBB    UNBRK                           ; REPLACE OPCODES
     CA 6A   04    E4   0A25                BBSC    #V_ATBRK,(R10),PROCEED          ; CHECK FOR PROCEED
             B0    11   0A29                BRB     OUTPC                           ; DISPLAY INSTRUCTION AND GET COMMANDS
                        0A2B
```

```
                         0A2B            .SBTTL  UNBRK - RESTORE OPCODES FOR BREAKPOINTS
                         0A2B       ;
                         0A2B       ;       UNBRK
                         0A2B       ;
                         0A2B       UNBRK:
        51    09    D0   0A2B            MOVL    #NBRK+NTMPBRK,R1          ; TOTAL PERM & TEMPORARY BREAKPOINTS
 50   F6B1 CF41    D0   0A2E       10$:  MOVL    BRKADR[R1],R0             ; GET BREAKPOINT ADDRESS
              06    13   0A34            BEQL    20$                       ; SKIP IF NOT ENABLED
                         0A36            .IF     DF,SW_PROCESS             ;
                         0A36            PUSHR   #^M<R0,R1,R2,R3,R4,R5>    ; SAVE REGS FOR PROTECTION CHANGE
                         0A36            MOVL    R0,R4                     ; FORM INADR RANGE FOR SET PROTECTION
                         0A36            MOVL    R0,R5                     ;
                         0A36            BSBW    SETWRT                    ; SET PAGE WRITABLE
                         0A36            MOVQ    (SP),R0                   ; RESTORE R0,R1
                         0A36            .ENDC                             ;
 60   F6D0 CF41    90   0A36            MOVB    BRKOP[R1],(R0)            ; RESTORE OPCODE
                         0A3C            .IF     DF,SW_PROCESS             ;
                         0A3C            BSBW    REPROT                    ; RESTORE PROTECTION
                         0A3C            POPR    #^M<R0,R1,R2,R3,R4,R5>    ; RESTORE REGISTERS
                         0A3C            .ENDC                             ;
           EF 51    F5   0A3C       20$:  SOBGTR  R1,10$                    ; DO THEM ALL
                    05   0A3F            RSB                               ; AND RETURN
                         0A40
```

```
                       0A40          .SBTTL  SETBRK - SET BREAK POINT INSTRUCTIONS
                       0A40     ;
                       0A40     ;            SETBRK
                       0A40     ;
         51    09  D0  0A40     SETBRK: MOVL    #NBRK+NTMPBRK,R1              ; TOTAL PERMANENT & TEMPORARY BRKPOINTS
 50  F69C CF41     D0  0A43     10$:    MOVL    BRKADR[R1],R0                 ; GET ADDRESS
               14  13  0A49             BEQL    20$                           ; SKIP IF NOT ENABLED
  F6BA CF41    60  90  0A4B             MOVB    (R0),BRKOP[R1]                ; SAVE OPCODE
         6A    18  93  0A51             BITB    #<<1@V_TBIT>!<1@V_ATBRK>>,(R10) ; CHECK FOR TRACE
               06  13  0A54             BEQL    15$                           ; NO TRACE, SET ANYWAY
      54 AB    50  D1  0A56             CMPL    R0,SAVPC-B(R11)               ; CHECK FOR AT BPT
               03  13  0A5A             BEQL    20$                           ; YES, DONT SET IT
                       0A5C     15$:                                          ;
                       0A5C             .IF     DF,SW_PROCESS                 ;
                       0A5C             PUSHR   #^M<R0,R1,R2,R3,R4,R5>        ; SAVE REGISTERS FOR PROTECTION CHANGE
                       0A5C             MOVL    R0,R4                         ; SET START ADDRESS OF RANGE
                       0A5C             MOVL    R0,R5                         ; AND END ADDRESS
                       0A5C             BSBW    SETWRT                        ;SET PAGE WRITABLE
                       0A5C             MOVL    (SP),R0                       ; RESTORE BPT ADDRESS
                       0A5C             .ENDC                                 ;
         60    03  90  0A5C             MOVB    #3,(R0)                       ; SET BREAKPOINT OPCODE
                       0A5F             .IF     DF,SW_PROCESS                 ;
                       0A5F             BSBW    REPROT                        ; RESTORE ORIGINAL PROTECTION VALUE
                       0A5F             POPR    #^M<R0,R1,R2,R3,R4,R5>        ; AND REGISTERS
                       0A5F             .ENDC                                 ;
      E1 51        F5  0A5F     20$:    SOBGTR  R1,10$                        ; DO THEM ALL
                   05  0A62             RSB                                   ; AND RETURN
                       0A63
```

```
                          0A63       .SBTTL  GETBPTX - GET INDEX FOR BREAKPOINT
                          0A63  ;
                          0A63  ;        GETBPTX
                          0A63  ;
                          0A63  GETBPTX:                                    ;
                53   08   D0  0A63          MOVL    #NBRK,R3                   ; INIT LOOP
F677 CF43       54   AB   D1  0A66  10$:    CMPL    SAVPC-B(R11),BRKADR[R3]    ; IS THIS A BPT?
                     03   13  0A6D          BEQL    20$                        ; YES
                F4   53   F5  0A6F          SOBGTR  R3,10$                     ; NO, CONTINUE
                     05       0A72  20$:    RSB                                ; RETURN
```

```
                 0A73        .SBTTL  QUOTE - INPUT CHARACTER STRING
                 0A73    ;
                 0A73    ;       QUOTE - START CHARACTER STRING INPUT
                 0A73    ;
                 0A73    QUOTE:                                  ;
    55   6B  D0  0A73            MOVL    CURDOT-B(R11),R5        ; POINT TO STRING BUFFER
       FAE1  30  0A76    5$:     BSBW    GETCHAR                 ; GET CHARACTER
    58   27  91  0A79            CMPB    #QUOT,R8                ; CHECK FOR QUOTE
         05  13  0A7C            BEQL    10$                     ; YES, END OF STRING
    85   58  90  0A7E            MOVB    R8,(R5)+                ; INSERT IN BUFFER
         F3  11  0A81            BRB     5$                      ; AND CONTINUE
    6B   55  D0  0A83    10$:    MOVL    R5,CURDOT-B(R11)        ; SAVE NEW DOT
             05  0A86            RSB                             ; RETURN
```

```
                    0A87       .SBTTL  DEPOSIT
                    0A87  ;
                    0A87  ;    DEPOSIT DATA
                    0A87  ;
                    0A87  DEPOSIT:
   1D 6A   1F   E0  0A87       BBS     #V_PREG,(R10),40$         ; BR IF PROCESSOR REGISTER
                    0A8B       .IF     DF,SW_PROCESS
                    0A8B       MOVL    CURDOT-B(R11),R4          ; GET CURRENT DOT
                    0A8B       TSTL    PID-B(R11)                ; CHECK FOR ARBITRARY PROCESS DEPOSIT
                    0A8B       BNEQ    50$                       ; BR IF YES
                    0A8B       .ENDC                             ;
                    0A8B       CASE    CURTYPE-B(R11),TYPE=B,<-        ; SWITCH ON TYPE
                    0A8B               10$,-                     ; BYTE
                    0A8B               20$,-                     ; WORD
                    0A8B               30$,-                     ; LONG
                    0A8B               >                         ;
                    0A96       .IF     NDF,SW_PROCESS            ;
00 BB   D8 AB   90  0A96  10$: MOVB    F1-B(R11),@CURDOT-B(R11); STORE BYTE
                05  0A9B       RSB                               ; RETURN
00 BB   D8 AB   B0  0A9C  20$: MOVW    F1-B(R11),@CURDOT-B(R11); STORE WORD
                05  0AA1       RSB                               ; RETURN
00 BB   D8 AB   D0  0AA2  30$: MOVL    F1-B(R11),@CURDOT-B(R11); STORE LONG
                05  0AA7       RSB                               ; RETURN
   6B   D8 AB   DA  0AA8  40$: MTPR    F1-B(R11),CURDOT-B(R11)   ; SET VALUE IN PROCESSOR REGISTER
                05  0AAC       RSB
                    0AAD       .IFF                              ; FALSE IF PROCESS VERSION
                    0AAD  10$:                                   ; BYTE DEPOSIT
                    0AAD       MOVL    R4,R5                     ; START AND END ADDRESSES EQUAL
                    0AAD       BSBW    SETWRT                    ; SET WRITABLE, OLD PROT TO R2
                    0AAD       MOVB    F1-B(R11),(R4)            ; STORE BYTE
                    0AAD       BSBW    REPROT                    ; RESTORE PROTECTION
                    0AAD       RSB                               ;
                    0AAD
                    0AAD  20$: ADDL3   #1,R4,R5                  ; WORD DEPOSIT, FORM END ADDRESS
                    0AAD       BSBW    SETWRT                    ; SET WRITABLE
                    0AAD       MOVW    F1-B(R11),(R4)            ; STORE WORD
                    0AAD       BSBW    REPROT                    ; RESTORE PROTECTION
                    0AAD       RSB                               ;
                    0AAD
                    0AAD  30$: ADDL3   #3,R4,R5                  ; LONGWORD DEPOSIT, FORM END ADDRESS
                    0AAD       BSBW    SETWRT                    ; SET WRITABLE
                    0AAD       MOVL    F1-B(R11),(R4)            ; STORE LONG WORD
                    0AAD       BSBW    REPROT                    ; RESTORE PROTECTION
                    0AAD       RSB                               ;
                    0AAD
                    0AAD  40$:                                   ; PROCESSOR REGISTER
                    0AAD       $CMKRNL_S       B^DEPPREG,(AP)    ; DEPOSIT IN PROCESSOR REGISTER
                    0AAD       RSB
                    0AAD  50$:                                   ; DEPOSIT IN ARBITRARY PROCESS
                    0AAD       CASE    CURTYPE-B(R11),TYPE=B,<-        ; SWITCH ON TYPE
                    0AAD               60$,-                     ; BYTE
                    0AAD               70$,-                     ; WORD
                    0AAD               80$>                      ; LONGWORD
                    0AAD       RSB
                    0AAD  60$: PUSHAB  W^DPBYTE                  ; SET ADDRESS OF BYTE ROUTINE
                    0AAD       BRB     90$                       ;
                    0AAD  70$: PUSHAB  W^DPWORD                  ; SET ADDRESS OF WORD ROUTINE
```

```
0AAD            BRB     90$                             ;
0AAD  80$:      PUSHAB  W^DPLONG                        ; SET ADDRESS OF LONG ROUTINE
0AAD  90$:      PUSHL   PID-B(R11)                      ; SET PID OF TARGET
0AAD            PUSHL   CURDOT-B(R11)                   ; ADDRESS FOR STORE
0AAD            PUSHL   F1-B(R11)                       ; VALUE TO STORE
0AAD            PUSHL   #4                              ; ARGUMENT COUNT
0AAD            MOVL    SP,R0                           ; POINTER TO ARGUMENT LIST
0AAD            TSTL    MFYFLG-B(R11)                   ; CHECK FOR STORE ENABLED
0AAD            BEQL    100$                            ; BR IF NOT
0AAD            $CMKRNL_S       W^QGET,(R0)             ; CALL TO QUEUE REQUEST
0AAD  100$:     ADDL    #20,SP                          ; CLEAN STACK
0AAD            RSB                                     ; AND RETURN
0AAD
0AAD  DEPPREG:.WORD     0                               ; DEPOSIT INTO PROCESSOR REGISTER
0AAD            MOVAB   W^PREXC,(FP)                    ; SET EXCEPTION HANDLER
0AAD            MTPR    F1-B(R11),CURDOT-B(R11)         ; PLACE FIELD VALUE IN REG
0AAD            MOVL    #1,R0                           ; RETURN SUCESS
0AAD            RET                                     ;
0AAD
0AAD  PREXC:    .WORD   0                               ; PROCESSOR REGISTER EXCEPTION HANDLER
0AAD            ADDL3   #4,8(AP),R1                     ; POINT TO EXCEPTION FP
0AAD            MOVL    (R1),12(FP)                     ; SET AS RETURN FP
0AAD            MOVAB   B^10$,16(FP)                    ; SET RETURN ADDRESS
0AAD  10$:      MOVZWL  #1,R0           ;SET NORMAL STATUS
0AAD            RET                                     ; AND RETURN
0AAD
0AAD            .ENDC
```

```
                         0AAD           .SBTTL  EXECUTE - PERFORM COMMAND STRING
                         0AAD      ;
                         0AAD      ;            EXECUTE
                         0AAD      ;
                         0AAD      EXECUTE:                                         ;
09 6A   08       E1      0AAD           BBC     #V_F1,(R10),10$              ; EXIT IF NO ADDRESS
59   D8 AB       D0      0AB1           MOVL    F1=B(R11),R9                 ; SET CHAR STRING
        03       12      0AB5           BNEQ    10$                          ; NOT NULL
      F723       31      0AB7           BRW     SUPERST                      ; SUPER RESET
                 05      0ABA      10$: RSB                                  ; RETURN
                         0ABB
```

```
                    0ABB          .SBTTL  P - PROCESSOR REGISTER PREFIX
                    0ABB     ;
                    0ABB     ;    SET PROCESSOR REGISTER MODE
                    0ABB     ;
                    0ABB     PREG:                              ; PROCESSOR REGISTER MODE
00 6A    0F     E2  0ABB          BBSS    #V_PRMODE,(R10),10$   ; SET PROCESSOR REG FLAG
                05  0ABF     10$: RSB                           ; RETURN
```

```
0AC0            .SBTTL  PROCESS DEBUGGER INITIALIZATION
0AC0
0AC0            .IF     DF,SW_PROCESS                           ;
0AC0    SALUTE: .ASCIZ  <CR><LF>/DELTA Version X2.2/<CR><LF>    ;
0AC0
0AC0    TEST:                                   ; START ADDRESS OF IMAGE ENTRY
0AC0    XDT$START::                             ; GLOBAL START ADDRESS FOR CLI DEBUG
0AC0            .WORD   0
0AC0    DELTA_START:                            ; START ADDRESS FOR DEBUGGER ENTRY
0AC0            $WAKE_S                         ; NULL WAKE AND
0AC0            $HIBER_S                        ; HIBERNATE TO GET SYNCHRONIZED
0AC0            MOVAB   TERMASK,TERMASKADR      ; RELOCATE TERMINATOR MASK DESCR
0AC0            MOVAB   TTNAMD+8,TTNAMD+4       ; RELOCATE DESCRIPTOR
0AC0            MOVAB   DBGINPUT+8,DBGINPUT+4
0AC0            MOVAB   TRNINPUT+8,TRNINPUT+4
0AC0            MOVAB   EXIHANDLE,EXIHADR       ;
0AC0            MOVAB   EXITCODE,EXCODA         ; RELOCATE EXIT HANDLER ARGS
0AC0            CALLG   (AP),B^INITCALL         ; GENERATE CALL FRAME
0AC0            RET                             ;
0AC0
0AC0    NOBRK:  MOVL    4(AP),AP                ; GET EXCEPTION ARGUMENT LIST
0AC0            BRW     EXCEPT+2                ; AND GOTO EXCEPTION HANDLER
0AC0
0AC0    INITCALL:
0AC0            .WORD   0                       ; ENTRY MASK
0AC0            MOVAB   W^CATCHALL,(FP)         ; SET CATCHALL EXCEPTION HANDLER
0AC0            $CMKRNL_S       W^SETKEXC,(AP)  ; SET EXCEPTION VECTORS FROM KERNEL MODE
0AC0            BLBS    R0,1$                   ; SUCCESS WITH KERNEL, IF NOT, TRY EXEC
0AC0            $CMEXEC_S       W^SETEEXC,(AP)  ; SET EXEC & SUPV EXCEPTION VECTORS
0AC0            BLBS    R0,1$                   ; BRANCH IF SUCCESS
0AC0            CMPL    R0,#SS$_NOPRIV          ; CHECK FOR LACK OF PRIVILEGE
0AC0            BEQL    1$                      ; WHICH IS THE ONLY ACCEPTABLE ERROR
0AC0            RET                             ; OTHERWISE GET OUT
0AC0    1$:     $SETEXV_S       ADDRES=W^EXCEPT,-       ;
0AC0                            ACMODE=#PSL$C_USER,-    ;
0AC0                            VECTOR=#0       ; SET PRIMARY FOR USER
0AC0            $SETEXV_S       ADDRES=W^CATCHALL,-     ; SET LAST CHANCE HANDLER
0AC0                            ACMODE=#PSL$C_USER,-    ; FOR USER MODE
0AC0                            VECTOR=#2       ; SPECIFY LAST CHANCE HANDLER
0AC0            $DCLEXH_S       EXITBLK         ; DECLARE USER MODE EXIT HANDLER
0AC0            PUSHR   #^M<R2,R3,R4,R5>
0AC0            MOVAB   W^DELBASE, R4           ; Set page protection on entire
0AC0            MOVAB   W^DELEND, R5            ; DELTA image to user writeable.
0AC0            BSBW    SETWRT
0AC0            POPR    #^M<R2,R3,R4,R5>
0AC0            $TRNLOG_S       LOGNAM=DBGINPUT,-       ; FIRST DEFAULT INPUT
0AC0                            RSLLEN=TRNINPUT,-
0AC0                            RSLBUF=TRNINPUT
0AC0            BLBC    R0,9$                   ; ON ERROR, USE TT
0AC0            CMPL    R0,#SS$_NOTRAN
0AC0            BEQL    9$                      ; USE TT IF NO TRANSLATION
0AC0    3$:     $TRNLOG_S       LOGNAM=TRNINPUT,-       ; TRANSLATE ALL THE WAY
0AC0                            RSLLEN=TRNINPUT,-
0AC0                            RSLBUF=TRNINPUT
0AC0            CMPL    R0,#SS$_NOTRAN
0AC0            BNEQ    3$                      ; TRY ANOTHER LEVEL
0AC0            CMPB    @TRNINPUT+4,#^X1B       ; CHECK FOR PROCESS PERMANENT
```

```
0AC0          BNEQ    5$
0AC0          SUBL    #4,TRNINPUT                  ; REMOVE THE ESCAPE HEADER
0AC0          ADDL    #4,TRNINPUT+4
0AC0  5$:     $ASSIGN_S       TRNINPUT,TTCHAN      ; DO THE ASSIGN
0AC0          BLBS    R0,10$                       ; TRY TT ON ERROR
0AC0  9$:     $ASSIGN_S       TTNAMD,TTCHAN        ; ASSIGN DEVICE
0AC0          BLBS    R0,10$                       ; CONTINUE IF SUCCESS
0AC0          RET                                  ; ELSE EXIT WITH ERROR CODE IN R0
0AC0  10$:    MOVAB   SALUTE,R4                    ; SET ADDRESS OF SALUTATION
0AC0          BSBW    OUTZSTRING                   ; OUTPUT IT
0AC0          BBC     #CLI$V_DBGEXCP,24(AP),15$    ; BR IF LATER INVOCATION
0AC0          BRW     NOBRK                        ; VIA $DEBUG COMMAND
0AC0  15$:    CALLG   (AP),B^20$                   ; CREATE TOP CALL FRAME
0AC0          RET                                  ;
0AC0  20$:    .WORD   0                            ; NULL ENTRY MASK
0AC0          ADDL    #4,4(AP)                     ; ADVANCE STARTING ADDRESS POINTER
0AC0          MOVPSL  -(SP)                        ; SAVE PSL
0AC0          ADDL3   #2,@4(AP),-(SP)              ; FETCH CURRENT STARTING ADDRESS
0AC0          MOVZWL  #SS$_DEBUG,-(SP)             ; SET EXCEPTION CODE
0AC0          PUSHL   #3                           ; SIGNAL ARG COUNT
0AC0          MOVL    SP,R0                        ; SAVE POINTER
0AC0          MOVQ    R0,-(SP)                     ; SAVE PHONY R0,R1
0AC0          PUSHL   #0                           ; DEPTH
0AC0          PUSHL   FP                           ; FP
0AC0          PUSHL   #4                           ; ARG COUNT
0AC0          PUSHL   SP                           ; POINTER TO MECH
0AC0          PUSHL   R0                           ; POINTER TO SIGNAL
0AC0          CALLS   #2,W^EXCEPT                  ; SIGNAL PHONY EXCEPTION
0AC0          ADDL    #12,SP                       ; CLEAN BACK TO R0,R1
0AC0          MOVQ    (SP)+,R0                     ; RESTORE R0,R1
0AC0          ADDL    #8,SP                        ; CLEAN BACK TO PC,PSL
0AC0          REI                                  ; RETURN TO TARGET PROGRAM
0AC0
0AC0
0AC0          .ENABLE         LOCAL_BLOCK
0AC0
0AC0  SETKEXC:        .WORD   0                    ; ENTRY MASK FOR CMKRNL PRIVILEGE
0AC0          PUSHAB  W^CLREXV_KERNEL              ; SET TO USE KERNEL RUNDOWN HANDLER
0AC0          CALLS   #1,W^SETRUNDWN               ; SET UP APPROPRIATE RUNDOWN HANDLER
0AC0          BLBC    R0,20$                       ; BRANCH IF CAN'T SET UP HANDLER
0AC0          $SETEXV_S       ADDRES=B^EXCEPT,-    ;
0AC0                          PRVHND=KCOND_PRIMARY,- ;
0AC0                          ACMODE=#PSL$C_KERNEL,- ; SET KERNEL
0AC0                          VECTOR=#0            ; PRIMARY VECTOR
0AC0          $SETEXV_S       ADDRES=W^CATCHALL,-  ;
0AC0                          PRVHND=KCOND_LASTCHANC,- ;
0AC0                          ACMODE=#PSL$C_KERNEL,- ; SET KERNEL MODE LAST CHANCE HANDLE
0AC0                          VECTOR=#2            ; SPECIFY LAST CHANCE VECTOR
0AC0          BRB     10$                          ; SKIP ALTERNATE ENTRY MASK
0AC0  ;--------------------------------------------------------------------------
0AC0  SETEEXC:        .WORD   0                    ; ENTRY MASK FOR CMEXEC PRIVILEGE
0AC0          PUSHAB  W^CLREXV_EXEC                ; SET TO USE EXEC RUNDOWN HANDLER
0AC0          CALLS   #1,W^SETRUNDWN               ; SET UP APPROPRIATE RUNDOWN HANDLER
0AC0          BLBC    R0,20$                       ; BRANCH IF CAN'T SET UP HANDLER
0AC0  10$:    $SETEXV_S       ADDRES=B^EXCEPT,-    ;
0AC0                          PRVHND=ECOND_PRIMARY,- ;
0AC0                          ACMODE=#PSL$C_EXEC,- ; SET EXEC MODE EXCEPTION HANDLER
```

```
0AC0                           VECTOR=#0                       ; PRIMARY VECTOR
0AC0            $SETEXV_S      ADDRES=W^CATCHALL,-             ;
0AC0                           PRVHND=ECOND_LASTCHANC,-        ;
0AC0                           ACMODE=#PSL$C_EXEC,-            ; SET EXEC MODE LAST CHANCE HANDLER
0AC0                           VECTOR=#2                       ; SPECIFY LAST CHANCE VECTOR
0AC0      ;--------------------------------------------------------------------------------
0AC0            $SETEXV_S      ADDRES=B^EXCEPT,-               ;
0AC0                           PRVHND=SCOND_PRIMARY,-          ;
0AC0                           ACMODE=#PSL$C_SUPER,-           ; SET SUPERVISOR MODE EXCEPTION HAND
0AC0                           VECTOR=#0                       ; PRIMARY VECTOR
0AC0            $SETEXV_S      ADDRES=W^CATCHALL,-             ;
0AC0                           PRVHND=SCOND_LASTCHANC,-        ;
0AC0                           ACMODE=#PSL$C_SUPER,-           ; SET SUPERVISOR LAST CHANCE HANDLER
0AC0                           VECTOR=#2                       ; SPECIFY LAST CHANCE VECTOR
0AC0      20$:  RET                                            ;
0AC0
0AC0            .DISABLE       LOCAL_BLOCK
0AC0
0AC0      EXCEPT: .WORD   0                                    ; EXCEPTION HANDLER ENTRY MASK
0AC0            $SETEXV_S      ADDRES=B^EXCEPT,-               ;
0AC0                           ACMODE=#PSL$C_USER,-            ;
0AC0                           VECTOR=#0                       ; RE-ESTABLISH USER PRIMARY VECTOR
0AC0            ADDL3   #4,4(AP),R0                            ; GET POINTER TO SIGNAL
0AC0            MOVPSL  R1                                     ; GET CURRENT PSL
0AC0            EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1      ;
0AC0            BBSS    R1,DBGACTIVE,40$                       ; BR IF ALREADY ACTIVE
0AC0            CMPL    #SS$_TBIT,(R0)                         ; IS IT TBIT?
0AC0            BNEQ    10$                                    ; NO,
0AC0      5$:   BRW     XDELTBIT                               ; YES, A TBIT
0AC0      10$:  CMPL    #SS$_BREAK,(R0)                        ; IS IT BREAKPOINT?
0AC0            BNEQ    20$                                    ; NO,
0AC0      15$:  BRW     XDELBPT                                ; YES, A BREAKPOINT
0AC0      20$:                                                 ; SOME OTHER EXCEPTION
0AC0            CMPL    #SS$_UNWINDING,(R0)                    ; IS IT UNWINDING
0AC0            BEQL    60$                                    ; YES
0AC0            CMPL    #SS$_COMPAT,(R0)+                      ; IS IT COMPATIBILITY MODE EXCEPT?
0AC0            BNEQ    30$                                    ; NO
0AC0            CMPL    #1,(R0)                                ; IS IT COMPATIBILITY BPT?
0AC0            BEQL    15$                                    ; YES
0AC0            CMPL    #7,(R0)                                ; IS IT COMPATIBILITY TBIT?
0AC0            BEQL    5$                                     ; YES
0AC0      30$:  CMPL    #SS$_DEBUG,-(R0)                       ; IS IT DEBUG EXCEPTION?
0AC0            BNEQ    40$                                    ; NO,
0AC0            BSBW    SAVE                                   ; SAVE EVERYTHING
0AC0            BRW     XDELDBG                                ; AND TREAT AS FUNNY BPT
0AC0      40$:                                                 ; UNEXPECTED EXCEPTION
0AC0            BBCC    R1,DBGACTIVE,50$                       ; CLEAR DEBUG ACTIVE
0AC0      50$:  CLRL    R0                                     ; RETURN FALSE FOR RESIGNAL
0AC0            RET
0AC0      60$:  MOVL    #1,R0                                  ; IGNORE AND RESIGNAL
0AC0            RET
0AC0            .PAGE
0AC0            .SBTTL  HANDLER FOR DEBUG EXCEPTIONS
0AC0
0AC0      DBGEXCEP:                                            ;
0AC0            .WORD   0
0AC0            ADDL3   #4,8(AP),R1                            ; POINT TO EXCEPTION FP
```

```
0AC0                 MOVL    FP,R0                        ; INIT LINK FOR CALL FRAMES
0AC0      10$:       CMPL    12(R0),(R1)                  ; IS THIS THE LAST ONE?
0AC0                 BEQL    20$                          ; YES
0AC0                 MOVAB   B^30$,16(R0)                 ; SET FOR RETURN
0AC0                 MOVL    12(R0),R0
0AC0                 BRB     10$                          ; CONTINUE
0AC0      20$:       MOVAB   XDELACV,16(R0)               ; SET RETURN FOR ERROR
0AC0      30$:       RET                                  ;
0AC0
0AC0      CATCHALL:                                       ; CATCHALL EXCEPTION HANDLER
0AC0                 .WORD   0                            ; ENTRY MASK
0AC0                 MOVPSL  R1                           ; GET CURMOD
0AC0                 EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1       ; ISOLATE CURRENT MODE
0AC0                 BBCS    R1,DBGACTIVE,10$             ; MUST NOT BE DEBUGGER EXCEPTION
0AC0                 CLRL    R0                           ; RESIGNAL
0AC0                 RET                                  ;
0AC0      10$:       BSBW    SAVE                         ; SAVE EVERYTHING
0AC0                 ADDL3   #4,4(AP),R0                  ; POINT TO EXCEPTION CODE
0AC0                 MOVL    (R0),R3                      ; GET IT
0AC0                 BSBW    CRLF                         ; OUTPUT CR/LF
0AC0                 BSBW    OUTLONG                      ; OUTPUT EXCEPTION CODE
0AC0                 MOVAB   B^EXCMSG,R4                  ; OUTPUT MESSAGE
0AC0                 BSBW    OUTZSTRING                   ; TEXT FOR EXCEPTION
0AC0                 BRW     XDELDBG                      ; AND DISPLAY INSTRUCTION
0AC0      EXCMSG: .ASCIZ     / EXCEPTION /                ;
0AC0
0AC0      EXIHANDLE:                                      ; EXIT HANDLER
0AC0                 .WORD   0                            ; ENTRY MASK
0AC0                 BITB    #15,DBGACTIVE                ; TEST FOR DEBUG ACTIVE IN ANY MODE
0AC0                 BEQL    10$                          ; NO, REPORT EXIT
0AC0                 RET                                  ; RETURN
0AC0      10$:                                            ; PROGRAM EXIT
0AC0                 MOVPSL  -(SP)                        ; BUILD EXCEPTION FRAME
0AC0                 PUSHL   16(FP)
0AC0                 PUSHL   @4(AP)                       ; EXIT CODE FOR EXCEPTION CODE
0AC0                 PUSHL   #3                           ; ARG COUNT
0AC0                 PUSHR   #^M<R0,R1>
0AC0                 MOVQ    AP,-(SP)
0AC0                 PUSHL   #4                           ; MECHANISM COUNT
0AC0                 PUSHL   SP                           ; POINTER TO MECHANISM
0AC0                 PUSHAL  24(SP)                       ; POINTER TO SIGNAL
0AC0                 PUSHL   #2
0AC0                 MOVL    SP,AP                        ; SET AP FOR EXCEPTION
0AC0                 BSBW    SAVE                         ; SAVE EVERYTHING
0AC0                 MOVAB   B^EXIMSG,R4                  ; DISPLAY EXIT MESSAGE
0AC0                 BSBW    OUTZSTRING                   ; OUTPUT TEXT
0AC0                 MOVL    SAVAP-B(R11),R3              ; GET POINTER TO EXCEPTION ARGLIST
0AC0                 MOVL    4(R3),R3                     ; GET EXIT CODE ADDRESS
0AC0                 BSBW    OUTLONG                      ; DISPLAY IT
0AC0                 $DCLEXH_S      EXITBLK               ; RE-ESTABLISH EXIT HANDLER
0AC0                 MOVPSL  R1                           ; GET CURRENT PSL
0AC0                 EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R1,R1 ; GET CURRENT MODE
0AC0                 BBSS    R1,DBGACTIVE,20$             ; SET DELTA ACTIVE FOR MODE
0AC0      20$:       BRW     XDELDBG                      ;
0AC0
0AC0      EXIMSG: .ASCIZ     <CR><LF>/ EXIT /             ;
0AC0
```

```
0AC0
0AC0            .ENABLE         LOCAL_BLOCK
0AC0    ;
0AC0    ; RESET INNER MODE EXCEPTION VECTORS. THIS CODE IS CALLED AS A
0AC0    ; PRIVILEGED IMAGE RUNDOWN HANDLER TO ENSURE ITS EXECUTION.
0AC0    ;
0AC0    ;  THIS ENTRY POINT IS USED IF THE PROCESS HAS CMKRNL PRIVILEGE
0AC0
0AC0    CLREXV_KERNEL:                          ; CLEAR EXCEPTION VECTORS
0AC0    ;
0AC0    ;  RESET PRIMARY AND LAST CHANCE EXCEPTION VECTORS FOR KERNEL MODE
0AC0    ;
0AC0            $SETEXV_S       ADDRES=@KCOND_PRIMARY,- ;
0AC0                            ACMODE=#PSL$C_KERNEL,-  ;
0AC0                            VECTOR=#0               ; PRIMARY VECTOR
0AC0            $SETEXV_S       ADDRES=@KCOND_LASTCHANC,-       ;
0AC0                            ACMODE=#PSL$C_KERNEL,-  ;
0AC0                            VECTOR=#2               ; LAST CHANCE VECTOR
0AC0
0AC0    ;  THIS ENTRY POINT IS USED IF THE PROCESS HAS CMEXEC PRIVILEGE
0AC0
0AC0    CLREXV_EXEC:                            ; CLEAR EXCEPTION VECTORS
0AC0    ;
0AC0    ;  RESET PRIMARY AND LAST CHANCE EXCEPTION VECTORS FOR EXECUTIVE MODE
0AC0    ;
0AC0            $SETEXV_S       ADDRES=@ECOND_PRIMARY,- ;
0AC0                            ACMODE=#PSL$C_EXEC,-    ;
0AC0                            VECTOR=#0               ; PRIMARY VECTOR
0AC0            $SETEXV_S       ADDRES=@ECOND_LASTCHANC,-       ;
0AC0                            ACMODE=#PSL$C_EXEC,-    ;
0AC0                            VECTOR=#2               ; LAST CHANCE VECTOR
0AC0    ;
0AC0    ;  RESET PRIMARY AND LAST CHANCE EXCEPTION VECTORS FOR SUPERVISOR MODE
0AC0    ;
0AC0            $SETEXV_S       ADDRES=@SCOND_PRIMARY,- ;
0AC0                            ACMODE=#PSL$C_SUPER,-   ;
0AC0                            VECTOR=#0               ; PRIMARY VECTOR
0AC0            $SETEXV_S       ADDRES=@SCOND_LASTCHANC,-       ;
0AC0                            ACMODE=#PSL$C_SUPER,-   ;
0AC0                            VECTOR=#2               ; LAST CHANCE VECTOR
0AC0            RSB                                     ;
0AC0
0AC0            .DISABLE        LOCAL_BLOCK
0AC0            .PAGE
0AC0            .SBTTL  SETRUNDWN - SET UP RUNDOWN HANDLER
0AC0
0AC0    ;++
0AC0    ;
0AC0    ; FUNCTIONAL DESCRIPTION:
0AC0    ;
0AC0    ;       This routine inserts the specified routine entry point into
0AC0    ;       the process' rundown vector. CMKRNL privilege of running in
0AC0    ;       kernel or exec mode is required.
0AC0    ;
0AC0    ; CALLING SEQUENCE:
0AC0    ;       CALLx SETRUNDWN
0AC0    ;
```

```
0AC0   ; INPUT PARAMETERS:
0AC0   ;       4(AP): address of rundown handler routine
0AC0   ;
0AC0   ; IMPLICIT INPUTS:
0AC0   ;       NONE
0AC0   ;
0AC0   ; OUTPUT PARAMETERS:
0AC0   ;       NONE
0AC0   ;
0AC0   ; IMPLICIT OUTPUTS:
0AC0   ;       NONE
0AC0   ;
0AC0   ; COMPLETION CODES:
0AC0   ;       SS$_NORMAL if successfully completed
0AC0   ;       SS$_NOPRIV if not privileged
0AC0   ;
0AC0   ; SIDE EFFECTS:
0AC0   ;       NONE
0AC0   ;
0AC0   ;--
0AC0
0AC0           .WEAK   CTL$GL_USRUNDWN
0AC0
0AC0   SETRUNDWN:
0AC0           .WORD   ^M<>
0AC0           MOVPSL  R0                          ; get PSL of caller
0AC0           CMPZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R0,#PSL$C_USER
0AC0           BLSSU   5$                          ; branch if DELTA started up in inner mode
0AC0           PUSHL   AP                          ; build $CMKRNL arg list on stack
0AC0           PUSHAB  10$                         ; address of kernel mode routine
0AC0           CALLS   #2,G^SYS$CMKRNL             ; call actual routine in kernel mode
0AC0           RET
0AC0
0AC0   5$:     MOVL    #SS$_NORMAL,R0              ; DELTA started up in an inner mode
0AC0           RET                                 ; do not set up a rundown handler
0AC0
0AC0   ;
0AC0   ; The following code executes in kernel mode and actually inserts the
0AC0   ; entry point into the rundown vector.
0AC0   ;
0AC0
0AC0   10$:    .WORD   ^M<>                        ; save no registers
0AC0           SETIPL  #IPL$_ASTDEL                ; lock out AST's while we modify the vector
0AC0           MOVAB   G^CTL$GL_USRUNDWN, R1       ; Get rundown vector pointer address.
0AC0           BEQL    19$                         ; Branch to NOP routine if no present.
0AC0           SUBL3   #4,(R1),R1                  ; Get address of rundown vector.
0AC0           MOVL    #SS$_VECFULL,R0
0AC0           CMPL    (R1),#<256-7>               ; check if another vector will fit
0AC0           BGEQU   20$                         ; branch if not
0AC0           ADDL3   (R1),R1,R0                  ; point to free vector
0AC0           MOVW    #^X9F16,(R0)+               ; insert JSB @#...
0AC0           MOVL    4(AP),(R0)+                 ; insert routine address
0AC0           MOVB    #^X05,(R0)+                 ; insert final RSB
0AC0           ADDL    #6,(R1)                     ; update end pointer
0AC0           ADDW    #6,G^IAC$AW_VECSET+4        ; adjust image activation end pointer
0AC0   19$:    MOVL    #SS$_NORMAL,R0              ; indicate success
0AC0   20$:    SETIPL  #0                          ; enable AST's again
```

```
0AC0           RET
0AC0           .PAGE
0AC0           .SBTTL  SETWRT - SET PAGES WRITABLE
0AC0   ;
0AC0   ; MAKE SPECIFIED RANGE OF PAGES WRITABLE
0AC0   ;
0AC0   ;       R4 = STARTING ADDRESS
0AC0   ;       R5 = ENDING ADDRESS
0AC0   ;
0AC0   ; R0-R2 DESTROYED
0AC0   ;
0AC0   ;
0AC0   SETWRT:                                          ;
0AC0           MOVQ    R4,-(SP)                         ; PUSH RANGE ONTO STACK
0AC0           MOVAL   -(SP),R2                         ; ADDRESS FOR RETURN OF PROT
0AC0           $CMKRNL_S B^SETPRTK,(R2)                 ; TRY IN KERNEL MODE FIRST
0AC0           BLBS    R0,10$                           ; CONTINUE IF NO ERROR
0AC0           CALLG   (R2),B^SETPRTK                   ; IF NOT ENOUGH PRIV, TRY USER MODE
0AC0   10$:    POPR    #^M<R2>                          ; RESTORE PROTECTION VALUE
0AC0           ADDL    #8,SP                            ; CLEAN STACK
0AC0           RSB                                      ; RETURN
0AC0
0AC0   SETPRTK:.WORD   0                                ;
0AC0           $SETPRT_S       INADR=4(AP),-            ;
0AC0                           PROT=#PRT$C_UW,-         ; WRITABLE BY ALL
0AC0                           ACMODE=#0,-              ;
0AC0                           PRVPRT=(AP)              ; ADDRESS AT WHICH TO RETURN PROT
0AC0           MOVL    #1,R0                            ; ALWAYS SUCCESS
0AC0           RET                                      ;
0AC0
0AC0   REPROT:                                          ; RESTORE PROTECTION
0AC0           RSB                                      ;
0AC0           .PAGE
0AC0           .SBTTL  FETCHP - FETCH DATA FROM ANOTHER PROCESS
0AC0
0AC0   FETCHP: CASE    CURTYPE-B(R11),TYPE=B,<-         ;
0AC0                   10$,-                            ; 0 => BYTE
0AC0                   20$,-                            ; 1 => WORD
0AC0                   30$>                             ; 2 => LONG
0AC0           RSB                                      ; UNKNOWN
0AC0   10$:    PUSHAB  W^FPBYTE                         ; SET FOR BYTE FETCH
0AC0           BRB     40$                              ;
0AC0   20$:    PUSHAB  W^FPWORD                         ; SET FOR WORD FETCH
0AC0           BRB     40$                              ;
0AC0   30$:    PUSHAB  W^FPLONG                         ; SET FOR LONGWORD FETCH
0AC0   40$:    PUSHL   PID-B(R11)                       ; PID OF TARGET PROCESS
0AC0           PUSHAB  QUAN-B(R11)                      ; SET ADDRESS TO RETURN VALUE
0AC0           PUSHL   CURDOT-B(R11)                    ; AND ADDRESS OF VALUE
0AC0           PUSHL   #4                               ; ARGUMENT COUNT
0AC0           MOVL    SP,R0                            ; SAVE POINTER TO ARG LIST
0AC0           $CMKRNL_S       W^QGET,(R0)              ; Q AST FOR DATA FETCH
0AC0           BLBC    R0,50$                           ; BR IF FAILED
0AC0           $HIBER_S                                 ; WAIT FOR DATA TO RETURN
0AC0   50$:    ADDL    #20,SP                           ; CLEAN STACK
0AC0           RSB                                      ; AND RETURN DATA
0AC0           .PAGE
0AC0           .SBTTL  QGET - QUEUE AST TO GET DATA FROM ANOTHER PROCESS
```

```
0AC0    ;
0AC0    ;       INPUTS: 04(AP) - LOCATION OF DATA
0AC0    ;               08(AP) - RETURN LOCATION
0AC0    ;               12(AP) - PID OF TARGET PROCESS
0AC0    ;               16(AP) - CODE SEGMENT POINTER
0AC0    ;
0AC0            .WEAK   EXE$ALLOCBUF                    ; MAKE FOLLOWING CODE OPTIONAL
0AC0            .WEAK   EXE$DEANONPAGED
0AC0            .WEAK   SCH$WAKE
0AC0            .WEAK   SCH$QAST
0AC0            .WEAK   SCH$GL_MAXPIX
0AC0
0AC0            FP_ORIGPID=ACB$L_AST
0AC0            FP_ADDR=ACB$L_ASTPRM
0AC0            FP_VALUE=ACB$L_ASTPRM
0AC0            FP_RETLOC=ACB$L_KAST+4
0AC0
0AC0    QGET:   .WORD   ^M<R2,R3,R4,R5>                 ; ENTRY MASK
0AC0            MOVZWL  #SS$_NONEXPR,R0                 ; ASSUME BAD PIX
0AC0            MOVAB   G^EXE$ALLOCBUF,R1               ; WERE WE LINKED WITH SYS.STB SYMBOLS?
0AC0            BEQL    10$                             ; IF NOT, RETURN WITH ERROR
0AC0            CMPW    12(AP),@#SCH$GL_MAXPIX          ; CHECK PIX FOR LEGAL PROCESS
0AC0            BGTRU   10$                             ; BR IF NOT
```

```
0AC0                    MOVZWL  @16(AP),R1                      ; GET SIZE OF CODE SEGMENT
0AC0                    MOVAB   IRP$C_LENGTH(R1),R1             ; ADD SIZE OF PACKET DATA
0AC0                    JSB     @#EXE$ALLOCBUF                  ; ALLOCATE BUFFER TO CONTAIN CODE
0AC0                    BLBC    R0,10$                          ; BRANCH IF NONE
0AC0                    MOVL    R2,R5                           ; SAVE ADDRESS OF PACKET
0AC0                    MOVL    PCB$L_PID(R4),FP_ORIGPID(R5)    ; SET PID FOR RETURN
0AC0                    MOVB    #^X80,ACB$B_RMOD(R5)            ; SET FOR SPECIAL KERNEL AST
0AC0                    MOVAB   ACB$L_KAST+8(R5),ACB$L_KAST(R5) ; SET ADDRESS FOR AST
0AC0                    MOVL    4(AP),FP_ADDR(R5)               ; SET ADDRESS FOR FETCH
0AC0                    MOVL    8(AP),FP_RETLOC(R5)             ; AND ADDRESS OF RETURN LOCATION
0AC0                    MOVL    16(AP),R0                       ; GET ADDRESS OF CODE SEGMENT
0AC0                    MOVL    12(AP),ACB$L_PID(R5)            ; SET TARGET PID
0AC0                    PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; SAVE REGS FOR MOVC
0AC0                    MOVC3   (R0)+,(R0),ACB$L_KAST+8(R5); COPY CODE SEGMENT TO BUFFER
0AC0                    POPR    #^M<R0,R1,R2,R3,R4,R5>          ; RESTORE REGISTERS
0AC0                    MOVZBL  #PRI$_TICOM,R2                  ; SET PRIORITY INCREMENT CLASS
0AC0                    JSB     @#SCH$QAST                      ; QUEUE AST FOR TARGET
0AC0          10$:      RET                                     ; RETURN TO ORIGINAL MODE
0AC0
0AC0                    .SBTTL  FPBYTE - FETCH BYTE FROM PROCESS
0AC0          FPBYTE:   .WORD   90$-.-2                         ; SIZE OF CODE SEGMENT
0AC0                    IFNORD  #1,@FP_ADDR(R5),10$             ; BRANCH IF NOT READABLE
0AC0                    MOVB    @FP_ADDR(R5),FP_VALUE(R5)       ; GET VALUE
0AC0          10$:      MOVL    FP_ORIGPID(R5),ACB$L_PID(R5) ; SET PID FOR RETURN AST
0AC0                    MOVB    #^X80,ACB$B_RMOD(R5)            ; SET FOR KAST AGAIN
0AC0                    MOVAB   B^20$,ACB$L_KAST(R5)            ; SET NEW AST ADDRESS
0AC0                    MOVZBL  #PRI$_TICOM,R2                  ; SET PRIORITY INCREMENT CLASS
0AC0                    JMP     @#SCH$QAST                      ; QUEUE RETURN AST
0AC0          20$:      IFNOWRT #1,@FP_RETLOC(R5),30$           ; IF NOT WRITABLE THEN SKIP IT
0AC0                    MOVB    FP_VALUE(R5),@FP_RETLOC(R5)     ; RETURN VALUE
0AC0          30$:      MOVL    ACB$L_PID(R5),R1                ; GET PID FOR WAKE
0AC0                    SETIPL  #IPL$_SYNCH                     ; RAISE TO SYNCH
0AC0                    JSB     @#SCH$WAKE                      ; WAKE PROCESS
0AC0                    SETIPL  #IPL$_ASTDEL                    ; LOWER IPL
0AC0                    MOVL    R5,R0                           ; SET ADDRESS FOR RELEASE
0AC0                    JMP     @#EXE$DEANONPAGED               ; FREE BLOCK AND EXIT
0AC0          90$:                                              ; END OF CODE SEGMENT
0AC0
0AC0                    .PAGE
0AC0                    .SBTTL  DPBYTE - DEPOSIT BYTE TO PROCESS
0AC0          DPBYTE:   .WORD   90$-.-2                         ; SIZE OF CODE SEGMENT
0AC0          20$:      IFNOWRT #1,@FP_RETLOC(R5),30$           ; IF NOT WRITABLE THEN SKIP IT
0AC0                    MOVB    FP_VALUE(R5),@FP_RETLOC(R5)     ; RETURN VALUE
0AC0          30$:      MOVL    R5,R0                           ; SET ADDRESS FOR RELEASE
0AC0                    JMP     @#EXE$DEANONPAGED               ; FREE BLOCK AND EXIT
0AC0          90$:                                              ; END OF CODE SEGMENT
0AC0
0AC0                    .PAGE
0AC0                    .SBTTL  FPWORD - FETCH WORD FROM PROCESS
0AC0          FPWORD:   .WORD   90$-.-2                         ; SIZE OF CODE SEGMENT
0AC0                    IFNORD  #2,@FP_ADDR(R5),10$             ; BRANCH IF NOT READABLE
0AC0                    MOVW    @FP_ADDR(R5),FP_VALUE(R5)       ; GET VALUE
0AC0          10$:      MOVL    FP_ORIGPID(R5),ACB$L_PID(R5) ; SET PID FOR RETURN AST
0AC0                    MOVB    #^X80,ACB$B_RMOD(R5)            ; SET FOR KAST AGAIN
0AC0                    MOVAB   B^20$,ACB$L_KAST(R5)            ; SET FOR NEW AST ADDRESS
0AC0                    MOVZBL  #PRI$_TICOM,R2                  ; SET PRIORITY INCREMENT CLASS
0AC0                    JMP     @#SCH$QAST                      ; QUEUE RETURN AST
```

```
0AC0   20$:     IFNOWRT #2,@FP_RETLOC(R5),30$    ; IF NOT WRITABLE THEN SKIP IT
0AC0            MOVW    FP_VALUE(R5),@FP_RETLOC(R5)     ; RETURN VALUE
0AC0   30$:     MOVL    ACB$L_PID(R5),R1         ; GET PID FOR WAKE
0AC0            SETIPL  #IPL$_SYNCH              ; RAISE TO SYNCH
0AC0            JSB     @#SCH$WAKE               ; WAKE PROCESS
0AC0            SETIPL  #IPL$_ASTDEL             ; LOWER IPL
0AC0            MOVL    R5,R0                    ; SET ADDRESS FOR RELEASE
0AC0            JMP     @#EXE$DEANONPAGED        ; FREE BLOCK AND EXIT
0AC0   90$:                                      ; END OF CODE SEGMENT
0AC0
0AC0            .PAGE
0AC0            .SBTTL  DPWORD - DEPOSIT WORD TO PROCESS
0AC0   DPWORD:  .WORD   90$-.-2                  ; SIZE OF CODE SEGMENT
0AC0   20$:     IFNOWRT #2,@FP_RETLOC(R5),30$    ; IF NOT WRITABLE THEN SKIP IT
0AC0            MOVW    FP_VALUE(R5),@FP_RETLOC(R5)     ; RETURN VALUE
0AC0   30$:     MOVL    R5,R0                    ; SET ADDRESS FOR RELEASE
0AC0            JMP     @#EXE$DEANONPAGED        ; FREE BLOCK AND EXIT
0AC0   90$:                                      ; END OF CODE SEGMENT
0AC0
0AC0            .PAGE
0AC0            .SBTTL  FPLONG - FETCH LONG FROM PROCESS
0AC0   FPLONG:  .WORD   90$-.-2                  ; SIZE OF CODE SEGMENT
0AC0            IFNORD  #4,@FP_ADDR(R5),10$      ; BRANCH IF NOT READABLE
0AC0            MOVL    @FP_ADDR(R5),FP_VALUE(R5) ; GET VALUE
0AC0   10$:     MOVL    FP_ORIGPID(R5),ACB$L_PID(R5) ; SET PID FOR RETURN AST
0AC0            MOVB    #^X80,ACB$B_RMOD(R5)     ; SET FOR KAST AGAIN
0AC0            MOVAB   B^20$,ACB$L_KAST(R5)     ; SET NEW KAST ADDRESS
0AC0            CLRL    R2                       ; NULL PRIO INCR
0AC0            JMP     @#SCH$QAST               ; QUEUE RETURN AST
0AC0   20$:     IFNOWRT #4,@FP_RETLOC(R5),30$    ; IF NOT WRITABLE THEN SKIP IT
0AC0            MOVL    FP_VALUE(R5),@FP_RETLOC(R5)     ; RETURN VALUE
0AC0   30$:     MOVL    ACB$L_PID(R5),R1         ; GET PID FOR WAKE
0AC0            SETIPL  #IPL$_SYNCH              ; RAISE TO SYNCH
0AC0            JSB     @#SCH$WAKE               ; WAKE PROCESS
0AC0            SETIPL  #IPL$_ASTDEL             ; LOWER IPL
0AC0            MOVL    R5,R0                    ; SET ADDRESS FOR RELEASE
0AC0            JMP     @#EXE$DEANONPAGED        ; FREE BLOCK AND EXIT
0AC0   90$:                                      ; END OF CODE SEGMENT
0AC0
0AC0            .PAGE
0AC0            .SBTTL  DPLONG - DEPOSIT LONGWORD TO PROCESS
0AC0   DPLONG:  .WORD   90$-.-2                  ; SIZE OF CODE SEGMENT
0AC0   20$:     IFNOWRT #4,@FP_RETLOC(R5),30$    ; IF NOT WRITABLE THEN SKIP IT
0AC0            MOVL    FP_VALUE(R5),@FP_RETLOC(R5)     ; RETURN VALUE
0AC0   30$:     MOVL    R5,R0                    ; SET ADDRESS FOR RELEASE
0AC0            JMP     @#EXE$DEANONPAGED        ; FREE BLOCK AND EXIT
0AC0   90$:                                      ; END OF CODE SEGMENT
0AC0   DELEND:                                   ;
0AC0            .ENDC                            ;
0AC0   ;
0AC0   ;        NORMAL END STATEMENT WITHOUT START ADDRESS
0AC0   ;        USED TO ASSEMBLE XDELTA FOR EXEC DEBUGGING.
0AC0   ;
0AC0            .END
```

```
$$BASE            = 00000001                 EXE$TBIT             *********   X   02
$$DISPL           = 00000008                 EXECUTE                00000AAD R    02
$$GENSW           = 00000001                 F1                     00000058 R    02
$$HIGH            = 00000007                 F2                     0000005C R    02
$$LIMIT           = 00000006                 F3                     00000060 R    02
$$LOW             = 00000001                 F4                     00000064 R    02
$$MNSW            = 00000001                 F5                     00000068 R    02
$$MXSW            = 00000001                 FCTR                   0000007C R    02
ADD                 00000286 R    02         FETCH                  00000304 R    02
ASTEN               000000E0 R    02         GETBPTX                00000A63 R    02
B                   00000080 R    02         GETCHAR                0000055A R    02
BLANK               000005CC R    02         GETCMD                 000009EE R    02
BMSG                00000964 R    02         GETSCB                 00000952 R    02
BRKADR            = 000000E4 R    02         GLOBL                  00000246 R    02
BRKCOM            = 00000131 R    02         GO                     000007A1 R    02
BRKDSP            = 00000111 R    02         HIGH                   0000024C R    02
BRKOP             = 0000010B R    02         INBUF                  00000004 R    02
BRKPOINT            00000702 R    02         INFLD                  00000242 R    02
BSLSH             = 0000005C                 INI$BRK              *********   X   02
CLR_730             0000084B R    02         INI$RDONLY           *********   X   02
CLR_750             0000084B R    02         INI$WRITABLE         *********   X   02
CLR_780             0000083F R    02         INSBUF                 00000078 R    02
CLR_790             00000850 R    02         INSLEN                 00000074 R    02
CLR_END             00000868 R    02         INSTR                  00000607 R    02
CLR_UV1             0000084B R    02         LBRACKET               0000067F R    02
COLON               000007B2 R    02         LF                   = 0000000A
COMMA               000002E6 R    02         LIB$INS_DECODE         *********W GX  02
CON$GETCHAR         *********   X 02         LINEFEED               00000346 R    02
CON$OWNCTY          *********   X 02         LOCOUT                 0000034E R    02
CON$PUTCHAR         *********   X 02         LOCP                   00000604 R    02
CON$RELEASECTY      *********   X 02         LOCPROMPT              0000034B R    02
CONTEXT             00000000 R    02         MCHK                   00000824 R    02
CONTEXTSZ         = 000000E4                 MCHKSAV                00000195 R    02
CR                = 0000000D                 MFYFLG                 0000006C R    02
CRLF                00000550 R    02         MFYFLGS                000007BC R    02
CURDOT              00000080 R    02         MMG$PAGEFAULT        *********   X   02
CURTYPE             0000007E R    02         MODES                  0000067A R    02
DCOM                000001D2 R    02         MUL                    0000027E R    02
DEPOSIT             00000A87 R    02         NBRK                 = 00000008
DIV                 00000282 R    02         NEGATE                 000005D5 R    02
DOT                 000007C2 R    02         NEXTDOT                0000032A R    02
DQUOTE              0000028A R    02         NEXTLOC                00000349 R    02
DTYPE               0000007D R    02         NEXTP                  000001ED R    02
ENDEXPR             0000025B R    02         NMODES               = 00000005
ENDFIELD            000002E9 R    02         NPRIM                = 0000002C
EQL1                0000063B R    02         NSEC                 = 00000007
EQUALS              00000634 R    02         NTERM                = 0000000A
ERR2                00000677 R    02         NTMPBRK              = 00000001
ERR3                00000868 R    02         OP$_ACBD             = 0000006F
ERR4                000002C5 R    02         OP$_ACBF             = 0000004F
ERROR               000001D6 R    02         OP$_ACBG             = 00004FFD
ESCAP               000005F1 R    02         OP$_ACBH             = 00006FFD
EXE$ACVIOLAT        *********   X 02         OP$_ADDD2            = 00000060
EXE$BREAK           *********   X 02         OP$_ADDD3            = 00000061
EXE$GB_CPUTYPE      *********   X 02         OP$_ADDF2            = 00000040
EXE$GL_SCB          *********   X 02         OP$_ADDF3            = 00000041
EXE$ROPRAND         *********   X 02         OP$_ADDG2            = 000040FD
```

```
OP$_ADDG3                 = 000041FD        OP$_CVTWD                 = 0000006D
OP$_ADDH2                 = 000060FD        OP$_CVTWF                 = 0000004D
OP$_ADDH3                 = 000061FD        OP$_CVTWG                 = 00004DFD
OP$_ADDP4                 = 00000020        OP$_CVTWH                 = 00006DFD
OP$_ADDP6                 = 00000021        OP$_DIVD2                 = 00000066
OP$_ASHP                  = 000000F8        OP$_DIVD3                 = 00000067
OP$_CLRD                  = 0000007C        OP$_DIVF2                 = 00000046
OP$_CLRF                  = 000000D4        OP$_DIVF3                 = 00000047
OP$_CLRG                  = 0000007C        OP$_DIVG2                 = 000046FD
OP$_CLRH                  = 00007CFD        OP$_DIVG3                 = 000047FD
OP$_CMPD                  = 00000071        OP$_DIVH2                 = 000066FD
OP$_CMPF                  = 00000051        OP$_DIVH3                 = 000067FD
OP$_CMPG                  = 000051FD        OP$_DIVP                  = 00000027
OP$_CMPH                  = 000071FD        OP$_EDITPC                = 00000038
OP$_CMPP3                 = 00000035        OP$_EMODD                 = 00000074
OP$_CMPP4                 = 00000037        OP$_EMODF                 = 00000054
OP$_CRC                   = 0000000B        OP$_EMODG                 = 000054FD
OP$_CVTBD                 = 0000006C        OP$_EMODH                 = 000074FD
OP$_CVTBF                 = 0000004C        OP$_MATCHC                = 00000039
OP$_CVTBG                 = 00004CFD        OP$_MNEGD                 = 00000072
OP$_CVTBH                 = 00006CFD        OP$_MNEGF                 = 00000052
OP$_CVTDB                 = 00000068        OP$_MNEGG                 = 000052FD
OP$_CVTDF                 = 00000076        OP$_MNEGH                 = 000072FD
OP$_CVTDH                 = 000032FD        OP$_MOVD                  = 00000070
OP$_CVTDL                 = 0000006A        OP$_MOVF                  = 00000050
OP$_CVTDW                 = 00000069        OP$_MOVG                  = 000050FD
OP$_CVTFB                 = 00000048        OP$_MOVH                  = 000070FD
OP$_CVTFD                 = 00000056        OP$_MOVP                  = 00000034
OP$_CVTFG                 = 000099FD        OP$_MOVTC                 = 0000002E
OP$_CVTFH                 = 000098FD        OP$_MOVTUC                = 0000002F
OP$_CVTFL                 = 0000004A        OP$_MULD2                 = 00000064
OP$_CVTFW                 = 00000049        OP$_MULD3                 = 00000065
OP$_CVTGB                 = 000048FD        OP$_MULF2                 = 00000044
OP$_CVTGF                 = 000033FD        OP$_MULF3                 = 00000045
OP$_CVTGH                 = 000056FD        OP$_MULG2                 = 000044FD
OP$_CVTGL                 = 00004AFD        OP$_MULG3                 = 000045FD
OP$_CVTGW                 = 000049FD        OP$_MULH2                 = 000064FD
OP$_CVTHB                 = 000068FD        OP$_MULH3                 = 000065FD
OP$_CVTHD                 = 0000F7FD        OP$_MULP                  = 00000025
OP$_CVTHF                 = 0000F6FD        OP$_POLYD                 = 00000075
OP$_CVTHG                 = 000076FD        OP$_POLYF                 = 00000055
OP$_CVTHL                 = 00006AFD        OP$_POLYG                 = 000055FD
OP$_CVTHW                 = 000069FD        OP$_POLYH                 = 000075FD
OP$_CVTLD                 = 0000006E        OP$_SCANC                 = 0000002A
OP$_CVTLF                 = 0000004E        OP$_SKPC                  = 0000003B
OP$_CVTLG                 = 00004EFD        OP$_SPANC                 = 0000002B
OP$_CVTLH                 = 00006EFD        OP$_SUBD2                 = 00000062
OP$_CVTLP                 = 000000F9        OP$_SUBD3                 = 00000063
OP$_CVTPL                 = 00000036        OP$_SUBF2                 = 00000042
OP$_CVTPS                 = 00000008        OP$_SUBF3                 = 00000043
OP$_CVTPT                 = 00000024        OP$_SUBG2                 = 000042FD
OP$_CVTRDL                = 0000006B        OP$_SUBG3                 = 000043FD
OP$_CVTRFL                = 0000004B        OP$_SUBH2                 = 000062FD
OP$_CVTRGL                = 00004BFD        OP$_SUBH3                 = 000063FD
OP$_CVTRHL                = 00006BFD        OP$_SUBP4                 = 00000022
OP$_CVTSP                 = 00000009        OP$_SUBP6                 = 00000023
OP$_CVTTP                 = 00000026        OP$_TSTD                  = 00000073
```

```
OP$_TSTF                  = 00000053
OP$_TSTG                  = 000053FD
OP$_TSTH                  = 000073FD
OPEN                        00000294 R     02
OPER                        0000007F R     02
OPERATOR                    000005CC R     02
OPERBAS                   = 00000012
OUTB                      = 00000006
OUTBB                       00000343 R     02
OUTBSLSH                    0000052B R     02
OUTBUF                      00000088 R     02
OUTCHAR                     00000534 R     02
OUTCOM                      00000507 R     02
OUTCR                     = 00000004
OUTDIGIT                    00000500 R     02
OUTER                       000001CA R     02
OUTINS                      00000383 R     02
OUTLONG                     00000504 R     02
OUTPC                       000009DB R     02
OUTPUT                      00000357 R     02
OUTPUTA                     000004A2 R     02
OUTPUT_ADDRESS              000003FE R     02
OUTRB                       00000531 R     02
OUTSPACE                    0000054B R     02
OUTZBUF                     0000051D R     02
OUTZSTRING                  00000521 R     02
OVEROPCLEN                = 00000005
OVEROPCODES                 00000199 R     02
OVRADR                      00000108 R     02
OVROPC                      000000E4 R     02
PFN$AB_STATE                ********   X   02
PFN$AB_TYPE                 ********   X   02
PFN$AL_BAK                  ********   X   02
PFN$AL_PTE                  ********   X   02
PFN$AW_REFCNT               ********   X   02
PFN$AW_SWPVBN               ********   X   02
PFN$AX_BLINK                ********   X   02
PFN$AX_FLINK                ********   X   02
PID                         00000070 R     02
PR$_EHSR                  = 0000004A
PR$_IPL                   = 00000012
PR$_KSP                   = 00000000
PR$_MAPEN                 = 00000038
PR$_MCESR                 = 00000026
PR$_SBIFS                 = 00000030
PR$_SCBB                  = 00000011
PR$_SID_TYP730            = 00000003
PR$_SID_TYP750            = 00000002
PR$_SID_TYP780            = 00000001
PR$_SID_TYP790            = 00000004
PR$_SID_TYPUV1            = 00000007
PRET                        00000253 R     02
PREG                        00000ABB R     02
PRIMARY                     0000019E R     02
PROCED                      000007AA R     02
PROCEED                     000009F3 R     02
PROGCTR                     000007D5 R     02
PSL$S_CURMOD              = 00000002
PSL$V_CURMOD              = 00000018
PSL$V_TBIT                = 00000004
QUAN                        00000084 R     02
QUANT                       000007CF R     02
QUOT                      = 00000027
QUOTE                       00000A73 R     02
RDBUF                     = 00000002
RDCR                      = 00000000
REGCOM                      000007EF R     02
REGISTER                    000007E7 R     02
RELOC                       00000472 R     02
RESET                       00000643 R     02
RESTORE                     000008EE R     02
RESTORR                     000008F3 R     02
RETURN                      000002C8 R     02
RSET                        000002DC R     02
RUBOUT                    = 0000007F
SAVAP                       000000C8 R     02
SAVE                        0000086B R     02
SAVOCR                      000000DC R     02
SAVPC                       000000D4 R     02
SAVPSL                      000000D8 R     02
SAVR2                       000000A0 R     02
SAVRCR                      000000DE R     02
SAVREG                      00000098 R     02
SAVRXCS                     000000E0 R     02
SAVSP                       000000D0 R     02
SCANP                       000001E9 R     02
SCH$GL_CURPCB               ********   X   02
SCH$GL_PCBVEC               ********   X   02
SECOND                      00000650 R     02
SEMI                        00000657 R     02
SETBRK                      00000A40 R     02
SHFT                        00000279 R     02
SHOBRK                      0000075F R     02
SLASH                       0000028F R     02
SLSH                      = 0000002F
SPACES                      0000037F R     02
STATUS                      00000054 R     02
STEP                        000006AE R     02
STEPOVER                    000006BB R     02
SUPERST                     000001DD R     02
SYSL$CLRSBIA                ********   X   02
TAB                         000005E1 R     02
TERM                        000001B7 R     02
UNBRK                       00000A2B R     02
VALI                        000007E4 R     02
VALR                        000007E1 R     02
VALUE                       000007D9 R     02
V_ASCII                   = 00000001
V_ATBRK                   = 00000004
V_F1                      = 00000008
V_F2                      = 00000009
V_F3                      = 0000000A
V_F4                      = 0000000B
V_F5                      = 0000000C
```

```
V_INFIELD                        = 00000002
V_INSTR                          = 0000000D
V_NEGATE                         = 00000007
V_OPEN                           = 00000000
V_PREG                           = 0000001F
V_PRMODE                         = 0000000F
V_RUB                            = 00000006
V_TBIT                           = 00000003
V_TBITOK                         = 00000005
XDELACV                            00000824 R      02
XDELBPT                            00000970 RG     02
XDELDBG                            00000A1F R      02
XDELIBRK                           000000E8 RG     02
XDELTBIT                           00000A08 RG     02
XDEL_LOADBASE                      00000161 RG     02
XDS$GL_XESTRING                    0000018D RG     02
XDS$GL_XFSTRING                    00000191 RG     02
XDS$GT_WORD_PFN                    ********   X    02
XREG                               00000815 R      02
XREGV                              00000155 R      02
XSET                               00000803 R      02

                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
Z$DEBUG_CODE                     00000AC0  ( 2752.)  02 (  2.)    PIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG

                                           +------------------------+
                                           ! Performance indicators !
                                           +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     30    00:00:00.09    00:00:01.55
Command processing                122    00:00:00.60    00:00:03.43
Pass 1                            694    00:00:20.73    00:01:22.10
Symbol table sort                   6    00:00:02.27    00:00:05.82
Pass 2                            414    00:00:06.03    00:00:28.24
Symbol table output                 2    00:00:00.18    00:00:00.32
Psect synopsis output               2    00:00:00.01    00:00:00.01
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             1272    00:00:29.92    00:02:01.49

The working set limit was 2250 pages.
144306 bytes (282 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 2050 non-local and 118 local symbols.
5124 source lines were read in Pass 1, producing 21 object records in Pass 2.
160 pages of virtual memory were used to define 157 macros.
```

```
                                    +--------------------------+
                                    ! Macro library statistics !
                                    +--------------------------+

Macro library name                  Macros defined
------------------                  --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                 12
_$255$DUA28:[SYSLIB]STARLET.MLB;2              11
TOTALS (all libraries)                         23

2055 GETS were required to define 23 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:XDELTA/OBJ=OBJ$:XDELTA MASD$:[EMULAT.SRC]MISSING/UPDATE=(MASD$:[EMULAT.ENH]MISSING)+MASD$:[DELTA.SRC]XDELTA/UPDATE=(M
```

DIFDEF
MDL

DIF

DIFF
MAP

XDSTRING
LIS

DIFPRE
REQ

DIFGETCMD
LIS

DIFHEXOCT
LIS

DATA
LIS

XDELTA
LIS