```
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO    TTTTTTTTTTTTTTT     SSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO    TTTTTTTTTTTTTTT     SSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO    TTTTTTTTTTTTTTT     SSSSSSSSSSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO       TTT            SSSSSSSSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO       TTT            SSSSSSSSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO       TTT            SSSSSSSSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO         TTT         SSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO         TTT         SSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO         TTT         SSSSSSSSSSS
```

```
BBBBBBBB      000000      000000    TTTTTTTTTT  5555555555    888888
BBBBBBBB      000000      000000    TTTTTTTTTT  5555555555    888888
BB      BB  00      00  00      00      TT      55            88    88
BB      BB  00      00  00      00      TT      55            88    88
BB      BB  00      00  00      00      TT      555555        88    88
BB      BB  00      00  00      00      TT      555555        88    88
BBBBBBBB    00      00  00      00      TT            55      888888
BBBBBBBB    00      00  00      00      TT            55      888888
BB      BB  00      00  00      00      TT              55    88    88
BB      BB  00      00  00      00      TT              55    88    88
BB      BB  00      00  00      00      TT      55      55    88    88      ....
BB      BB  00      00  00      00      TT      55      55    88    88      ....
BBBBBBBB      000000      000000        TT        555555      888888        ....
BBBBBBBB      000000      000000        TT        555555      888888        ....

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     1          .TITLE  BOOT58 - Bootstrap command processor
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28  ;++
0000    29  ;
0000    30  ; FACILITY:
0000    31  ;
0000    32  ;       Bootstrap command processing module
0000    33  ;
0000    34  ; ENVIRONMENT:
0000    35  ;
0000    36  ;       Runs at IPL 31, kernel mode, memory management is OFF, IS=1
0000    37  ;       (running on interrupt stack), and code must be PIC.
0000    38  ;
0000    39  ; ABSTRACT:
0000    40  ;
0000    41  ;       This module gains control from the boot block 0 code. The module
0000    42  ;       processes commands entered interactively from the console
0000    43  ;       terminal, or commands contained in command procedure files on
0000    44  ;       the boot device.
0000    45  ;
0000    46  ; AUTHOR:
0000    47  ;
0000    48  ;       Carol Peters    22 March 1979
0000    49  ;
0000    50  ; MODIFIED BY:
0000    51  ;
0000    52  ;       V03-003 ACG0392         Andrew C. Goldstein,    19-Jan-1984  22:39
0000    53  ;               Tie off SYS$FILESCAN for TPARSE use
0000    54  ;
0000    55  ;       V03-002 KTA3085         Kerbey T. Altmann       03-Oct-1985
0000    56  ;               Add some more dummy symbols for XDELTA.  Add ability
0000    57  ;               to boot unattended.
```

```
0000     58 ;
0000     59 ;       V03-001 ACG0345         Andrew C. Goldstein,    1-Aug-1983  16:45
0000     60 ;               Add dummy SYS$ASCTOID routine for TPARSE
0000     61 ;
0000     62 ;       Steve Jeffreys  04-DEC-1979
0000     63 ;               - Added documentation for indirect command file processing.
0000     64 ;               - Echo command line for indirect command files.
0000     65 ;               - Added HELP command.
0000     66 ;               - Added BOOT command.
0000     67 ;               - Fixed D/I command.
0000     68 ;               - Added /B size qualifier.
0000     69 ;               - Make START command invalidate the Translation Buffer
0000     70 ;                 before JMPing to the new image.
0000     71 ;--
```

```
                        0000     73          .SBTTL  Declarations
                        0000     74
                        0000     75 ;
                        0000     76 ; Macros
                        0000     77 ;
                        0000     78
                        0000     79          $DSCDEF                         ; String descriptor definitions
                        0000     80          $PRDEF                          ; Processor registers
                        0000     81          $RPBDEF                         ; RPB definitions
                        0000     82          $SSDEF                          ; Status code definitions
                        0000     83          $TPADEF                         ; TPARSE field definitions
                        0000     84
                        0000     85 ;
                        0000     86 ; Macro to print messages to terminal.
                        0000     87 ;
                        0000     88
                        0000     89          .MACRO  MSG,STR
                        0000     90          BSBW    BOO$FACMSG
                        0000     91          .ASCIZ  \'STR'\
                        0000     92          .ENDM   MSG
                        0000     93
                        0000     94 ;
                        0000     95 ; Equated symbols:
                        0000     96 ;
                        0000     97
0000000D                0000     98          CAR_RETURN      = 13            ; ASCII carriage return code.
0000000C                0000     99          FORM_FEED       = 12            ; ASCII form feed code.
0000000B                0000    100          VERTICAL_TAB    = 11            ; ASCII vertical tab.
0000000A                0000    101          LINE_FEED       = 10            ; ASCII linefeed code.
00000000                0000    102          NULL            = 0             ; ASCII null character code.
                        000C    103
00000001                0000    104          SADEBUG         = 1             ; Include XDELTA in image.
00000001                0000    105          SAENVIRON       = 1             ; Operating standalone.
00000000                0000    106          TSDEBUG         = 0             ; Include DEBUG in image.
                        0000    107
                        0000    108          _VIELD  CMD,0,<-                ; Command details bit defs.
                        0000    109                  <LOADCMD,,M>,-          ; Load command.
                        0000    110                  <STARTCMD,,M>,-         ; Start command.
                        0000    111                  <EXAMCMD,,M>,-          ; Examine command.
                        0000    112                  <DEPOSCMD,,M>,-         ; Deposit command.
                        0000    113                  <HELPCMD,,M>,-          ; Help command.
                        0000    114                  <BOOTCMD,,M>,-          ; Boot command.
                        0000    115                  <CMDFILCMD,,M>,-        ; Indirect command file.
                        0000    116                  <INDIRECT,,M>,-         ; Processing indirect file.
                        0000    117                  <PHYSICAL,,M>,-         ; Physical address.
                        0000    118                  <GENERAL,,M>,-          ; General register.
                        0000    119                  <INTERNAL,,M>,-         ; Internal processor register.
                        0000    120                  <LONG,,M>,-             ; Longword length field.
                        0000    121                  <WORD,,M>,-             ; Word length field.
                        0000    122                  <BYTE,,M>,-             ; Byte length field.
                        0000    123                  <NULLDEV,,M>,-          ; Null boot device
                        0000    124                  <UNATTEND,,M>,-         ; Unattended DEFBOO
                        0000    125                  >
                        0000    126
0000000B                0000    127          RPB$V_NODEFBOO  = RPB$V_MPM     ; Reuse this bit as no MA750 exists.
00000800                0000    128          RPB$M_NODEFBOO  = RPB$M_MPM
                        0000    129
```

```
00000007  0000   130          NUM_OF_CMDS     = 7             ; # of commands
0000003F  0000   131          MAX_PROC_REG    = ^X3F          ; Highest processor register.
          0000   132
00000000  0000   133          SAVED_R0        = 0             ; Offset to registers saved on
00000001  0000   134          SAVED_R1        = 1             ; stack.
00000002  0000   135          SAVED_R2        = 2
00000003  0000   136          SAVED_R3        = 3
00000004  0000   137          SAVED_R4        = 4
00000005  0000   138          SAVED_R5        = 5
00000006  0000   139          SAVED_R6        = 6
00000007  0000   140          SAVED_R7        = 7
00000008  0000   141          SAVED_R8        = 8
00000009  0000   142          SAVED_R9        = 9
0000000A  0000   143          SAVED_R10       = 10
0000000B  0000   144          SAVED_R11       = 11
0000000C  0000   145          SAVED_AP        = 12
0000000D  0000   146          SAVED_FP        = 13
0000000E  0000   147          SAVED_SP        = 14
0000000F  0000   148          SAVED_PC        = 15
          0000   149
          0000   150 ;
          0000   151 ; BOOT58 resides in the upper quarter of the 64K bytes of good memory
          0000   152 ; available during bootstrap operations. The BOOT58 location allows
          0000   153 ; VMB to be loaded at <base of memory + ^X200> as usual, and also
          0000   154 ; permits the diagnostic supervisor to be loaded at
          0000   155 ; <base of memory + ^XE000>. If BOOT58 grows in size, it may be
          0000   156 ; necessary to lower its address.
          0000   157 ;
          0000   158
0000C000  0000   159          BOOT58_OFFSET   = ^XC000        ; Offset to BOOT58 from start
          0000   160                                          ; of good memory.
          0000   161
          0000   162 ;
          0000   163 ; Own storage.
          0000   164 ;
          0000   165
        00000000 166          .PSECT  BOOT58_DATA,LONG        ; Data PSECT.
          0000   167
          0000   168 PARAM_BLOCK:                           ; Parser parameter block.
00000024  0000   169          .BLKB   TPA$K_LENGTH0
          0024   170
          0024   171 GOODMEM_BASE:                          ; Base address of good memory.
00000000  0024   172          .LONG   0
          0028   173
          0028   174 FIRST_FREE_BYTE:                       ; Address of first free byte
00000000  0028   175          .LONG   0                       ; past BOOT58.
          002C   176
          002C   177 COMMAND_FLAGS:                         ; Mask describing details of
00000000  002C   178          .LONG   0                       ; current and previous commands.
          0030   179
          0030   180 CURRENT_ADDRESS:                       ; Address field of this EXAMINE
00000000  0030   181          .LONG   0                       ; or DEPOSIT command.
          0034   182
          0034   183 LAST_ADDRESS:                          ; Address field of last EXAMINE
00000000  0034   184          .LONG   0                       ; or DEPOSIT command.
          0038   185
          0038   186 LAST_VALUE:                            ; Last value displayed.
```

```
                    00000000  0038   187          .LONG   0
                              003C   188
                              003C   189 SAVED_VALUE:                        ; Holds a value.
                    00000000  003C   190          .LONG   0
                              0040   191
                              0040   192 REGISTER_TABLE:                     ; Address of register values.
                    00000000  0040   193          .LONG   0
                              0044   194
                              0044   195 HOLD_DEPOSIT:                      ; Value to be deposited in
                    00000000  0044   196          .LONG   0                 ; DEPOSIT command.
                              0048   197
                              0048   198 START_ADDRESS:                     ; Address given in START or
                    00000000  0048   199          .LONG   0                 ; LOAD command.
                              004C   200
                              004C   201 FILE_DESCRIP:                      ; Descriptor of file spec.
          00000000 00000000   004C   202          .LONG   0,0
                              0054   203
                              0054   204 HELP_FILE_SPEC:                    ; Help file name string.
50 4C 48 2E 4C 4F 53 4E 4F 43 0054   205          .ASCII  /CONSOL.HLP/
                              005E   206
                              005E   207 HELP_FILE_DESCRIP:                ; File descripter for HELP file.
                    0000000A  005E   208          .LONG   10                ;       File spec length
                    00000000  0062   209          .LONG   0                 ;       File spec address
                              0066   210
                              0066   211 DEFBOO_SPEC:                       ; Default boot command file spec
44 4D 43 2E 4F 4F 42 46 45 44 0066   212          .ASCII /DEFBOO.CMD/       ;
                              0070   213
                              0070   214 DEFBOO_DESCRIP:                   ; Default boot command file descripter
                    0000000A  0070   215          .LONG   10                ;       File spec length
                    00000000  0074   216          .LONG   0                 ;       File spec address
                              0078   217
                              0078   218 BOOTFILE_SPEC:                     ; Boot command file spec
44 4D 43 2E 4F 4F 42 58 58 58 0078   219          .ASCII /XXXBOO.CMD/       ;
                              0082   220
                              0082   221 BOOTFILE_DESCRIP:                 ; Boot command file descripter
                    0000000A  0082   222          .LONG   10                ;       File spec length
                    00000000  0086   223          .LONG   0                 ;       File spec address
                              008A   224
                              008A   225 DEV_SPEC:                          ; Device spec
                    0000008F  008A   226          .BLKB   5                 ;       Allow only 5 characters
                              008F   227
                              008F   228 DEV_DESCRIP:                       ; Device descripter
                    00000000  008F   229          .LONG   0                 ;       Device spec length
                    00000000  0093   230          .LONG   0                 ;       Device spec address
                              0097   231
                              0097   232 UNIT_NUMBER:                       ; Boot device unit number
                    00000000  0097   233          .LONG   0                 ;       Values 0..9
                              009B   234
                              009B   235 FILE_SPEC:                         ; Address of file specification.
                    000000DB  009B   236          .BLKB   64
                              00DB   237
                              00DB   238 COMMAND_BUFFER:                    ; Command buffer.
                54 4F 4F 42 00' 00DB   239          .ASCIC  /BOOT/            ; Unattended DEFBOO
                          04    00DB
                    000001A3  00E0   240          .BLKB   200-5
                              01A3   241
                    000000C4  01A3   242 CMD_BUFFER_SIZE = .-COMMAND_BUFFER-4     ; Length of command buffer.
```

```
                        01A3   243                                     ; (Allow 4 bytes for LINE_TERM)
                        01A3   244
                        01A3   245 COMMAND_STRING:                     ; Command string buffer.
                        01A3   246 DIREC_BUFFER:                       ; RT-11 directory buffer.
                000005A3 01A3   247         .BLKB   1024
                        05A3   248
                00000400 05A3   249 CMD_STRING_SIZE = .-COMMAND_STRING ; Length of command string.
                        05A3   250
                        05A3   251 IND_CMDBUFFER:                      ; Indirect command file buffer.
                000007A3 05A3   252         .BLKB   512
                        07A3   253
                        07A3   254 IND_CMDBUF_PTR:                     ; Pointer to next byte to read
                00000000 07A3   255         .LONG   0                  ; in indirect command buffer.
                        07A7   256
                        07A7   257 IND_W_BLKS_LEFT:                    ; number of blocks left to read
                    0000 07A7   258         .WORD   0                  ; in command file
                        07A9   259
                        07A9   260 IND_W_NEXT_LBN:                     ; Next LBN to read from indirect
                    0000 07A9   261         .WORD   0                  ; command file.
                        07AB   262
                        07AB   263 SAVED_STATUS:                       ; Temp storage of exit status
                000007AD 07AB   264         .BLKW   1
                        07AD   265
                        07AD   266 IN_COMMAND_FILE:                    ; Used as boolean
                000007AE 07AD   267         .BLKB   1
                        07AE   268
                        07AE   269 PROMPT_STRING:                      ; BOOT58's prompt string.
      00 3E 38 35 54 4F 4F 42 0A 0D 07AE   270         .ASCIZ  <CAR_RETURN><LINE_FEED>/BOOT58>/
                        07B8   271
                        07B8   272 LINE_TERM:                          ; Line terminator for lines to be echoed.
                  00 00 0A 0D 07B8   273         .ASCIZ  <CAR_RETURN><LINE_FEED><NULL>
                        07BC   274
                        07BC   275 OUTPUT_BUFFER:                      ; FAO output buffer.
                   000007E4 07BC   276         .BLKB   40
                        07E4   277
                   00000028 07E4   278 OUTBUF_LENGTH   = .-OUTPUT_BUFFER ; Length of FAO output buffer.
                        07E4   279
                        07E4   280 W_FAO_OUT_LEN:                      ; Location to hold length of
                      0000 07E4   281         .WORD   0                  ; FAO output string.
                        07E6   282
                        07E6   283 LONG_PHY_FORMAT:                    ; FAO control string.
20 4C 58 38 21 20 20 50 5F 21 2F 21 07E6   284         .ASCII  %!/!_P  !8XL  !8XL%  ; Display a hex longword.
                  4C 58 38 21 20 07F2
                        07F7   285
                   00000011 07F7   286 FORMAT_LENGTH   = .-LONG_PHY_FORMAT ; Length of FAO format string.
                        07F7   287
                        07F7   288 OUTPUT_FORMAT:                      ; String in which to store a
                   00000808 07F7   289         .BLKB   FORMAT_LENGTH      ; dynamic output format.
                        0808   290
                        0808   291 B_ERROR_OUT:                        ; Error message already output
                      00 0808   292         .BYTE   0                  ; for this command.
                        0809   293
                        0809   294 B_ENVIRON_SAFE:                     ; Flag saying that an exception
                      00 0809   295         .BYTE   0                  ; occurred during parse.
                        080A   296
                        080A   297 L_OLD_STACKP:                       ; Location to save stack
                00000000 080A   298         .LONG   0                  ; pointer in case of exceptions.
```

```
                080E   299
                080E   300 DRIVER_SUBROUT::                    ; Holds address of device ROM
00000000        080E   301         .LONG   0                   ; driver subroutine.
                0812   302
                0812   303 ;
                0812   304 ; Declarations for the aid of XDELTA.
                0812   305 ;
                0812   306
            00000000   307         .PSECT  __Z99BOOT,Page      ; PSECT that always links at end
                0000   308                                     ; of this program.
                0000   309
                0000   310 BOOTHIGH:                           ; Symbol to mark the start of
                0000   311                                     ; the first page after the code
                0000   312                                     ; in this module.
                0000   313
00000001        0000   314         .IF     NE,SADEBUG          ; If XDELTA is included,
00000004        0000   315 EXE$MCHKVEC     == BOOTHIGH+4       ; define a symbol used by XDELTA
                0000   316         .ENDC                       ; to locate the SCB.
```

```
0000    318          .SBTTL  Parse tables
0000    319 ;
0000    320 ; The following documenation is taken from the help file that is printed
0000    321 ; by BOOT58 via the HELP command.
0000    322 ;
0000    323
0000    324 ;++
0000    325 ; GENERAL:
0000    326 ;
0000    327 ; <location>     ::= <value> ! <register>
0000    328 ;
0000    329 ; <value>        ::= <number> ! <shorthand>
0000    330 ;
0000    331 ; <number>       ::= Any nonnegative HEXADECIMAL number.
0000    332 ;
0000    333 ; <register>     ::= 0..F
0000    334 ;
0000    335 ; <shorthand>    ::= Any one of the following:
0000    336 ;                          "*" - use LAST <location> specified
0000    337 ;                          "+" - use (LAST <location>) + 1
0000    338 ;                          "-" - use (LAST <location>) - 1
0000    339 ;                          "@" - use contents of (LAST <location>)
0000    340 ;
0000    341 ; <loc_qual>     ::= Any one of the following:
0000    342 ;                          "/P" - physical memory address
0000    343 ;                          "/G" - general register
0000    344 ;                          "/I" - internal processor register
0000    345 ;
0000    346 ; <size_qual>    ::= Any one of the following:
0000    347 ;                          "/L" - size = longword
0000    348 ;                          "/W" - size = word
0000    349 ;                          "/B" - size = byte
0000    350 ;
0000    351 ; <devspec>      ::= A device spec of the form: DDCU, where
0000    352 ;                          DD = Generic device type (eg. DB)
0000    353 ;                          C  = controller designator (A ! B)
0000    354 ;                          U  = unit number (0..9)
0000    355 ;
0000    356 ; <filespec>     ::= A legal RT-11 filename of the form: NAME.TYP, where
0000    357 ;                          NAME = any alphanumeric string of not more than 6 chars.
0000    358 ;                          TYP  = any alphanumeric string of not more than 3 chars.
0000    359 ;                                 a null TYP is acceptable.
0000    360 ;
0000    361 ; BOOT58 COMMANDS:
0000    362 ;
0000    363 ; BOOT [<devspec>]
0000    364 ;
0000    365 ;       Boot from the device specified.  If no device is specified, boot from
0000    366 ;       the default boot device.  This command cannot be used within an indirect
0000    367 ;       command  file.
0000    368 ;
0000    369 ;
0000    370 ; DEPOSIT [<loc_qual><size_qual>] <location> <value>
0000    371 ;
0000    372 ;       Deposit <value> at the location specifed by <location>.
0000    373 ;       The <location> is interpreted according to the qualifiers.
0000    374 ;
```

```
0000   375 ;
0000   376 ; EXAMINE [<loc_qual><size_qual>] <location>
0000   377 ;
0000   378 ;       Display the contents of <location>, where <location> is interpreted
0000   379 ;       according to the qualifiers.
0000   380 ;
0000   381 ;
0000   382 ; HELP
0000   383 ;
0000   384 ;       Print this text at the console.  This command cannot be used within
0000   385 ;       an indirect command file.
0000   386 ;
0000   387 ; LOAD <filespec>[/START:<value>]
0000   388 ;
0000   389 ;       Load a file from the boot device into memory starting at the address
0000   390 ;       specified <number>.  If no starting location is specified, load the
0000   391 ;       file beginning at the first free location in memory.
0000   392 ;
0000   393 ;
0000   394 ; START <value>
0000   395 ;
0000   396 ;       JMP to <value>.
0000   397 ;
0000   398 ;
0000   399 ; NOTE:
0000   400 ;
0000   401 ;       All BOOT58 commands may be abbreviated to the least amount of
0000   402 ;       significant characters.  Since the first character of each command
0000   403 ;       is unique, this means all commands may be abbreviated to the one
0000   404 ;       character.  For example, the BOOT command may be entered as "B".
0000   405 ;
0000   406 ;       Some processor registers are either read or write only.  Attempting
0000   407 ;       the wrong operation (ie. E/I <write_only_reg>) on such a register
0000   408 ;       will yield unpredictable results.
0000   409 ;--
0000   410
0000   411
0000   412         $INIT_STATE -                           ; Start of TPARSE state table.
0000   413                 BOOT58_TABLE,KEY_TABLE  ; Name of state and keyword
0000   414                                                 ; table.
0000   415
0000   416 ;
0000   417 ; Define a command line.
0000   418 ;
0000   419
0000   420         $STATE
0000   421         $TRAN   !VALID_COMMAND,-                ; Accept valid command.
0000   422                 TPA$_EXIT,-
0000   423                 EXECUTE_COMMAND
0000   424         $TRAN   TPA$_EOS,TPA$_EXIT              ; Accept null command.
0000   425
0000   426 ;
0000   427 ; Parse a valid command, which is a syntactically-correct command
0000   428 ; followed immediately by end of line.
0000   429 ;
0000   430
0000   431         $STATE  VALID_COMMAND                   ; Accept valid command.
```

```
                                I 13
0000    432          $TRAN   !LOADCMD,,SET_LOAD_FLAG ; Set flag for load command.
0000    433          $TRAN   !STARTCMD,,-            ; Set flag for start command.
0000    434                  SET_START_FLAG
0000    435          $TRAN   !EXAMINECMD,,-          ; Set flag for examine command.
0000    436                  SET_EXAM_FLAG
0000    437          $TRAN   !DEPOSITCMD,,-          ; Set flag for deposit command.
0000    438                  SET_DEPOS_FLAG
0000    439          $TRAN   !HELPCMD,,SET_HELP_FLAG ; Set flag for help command.
0000    440          $TRAN   !BOOTCMD,,SET_BOOT_FLAG ; Set flag for boot command.
0000    441          $TRAN   !CMDFILE,,-             ; Set flag for indirect command
0000    442                  SET_CMDFIL_FLAG         ; file.
0000    443          $STATE
0000    444          $TRAN   TPA$_EOS,TPA$_EXIT      ; End command with end of line.
0000    445
0000    446 ;
0000    447 ; Load command.
0000    448 ;
0000    449
0000    450          $STATE  LOADCMD                 ; Command that loads a file
0000    451                                          ; into memory.
0000    452          $TRAN   'LOAD'                  ; Command verb.
0000    453          $STATE
0000    454          $TRAN   !FILESPEC,,MOV_FILESPEC ; Parse file specification.
0000    455          $STATE
0000    456          $TRAN   !LOAD_QUAL,TPA$_EXIT    ; Optional start address.
0000    457          $TRAN   TPA$_LAMBDA,TPA$_EXIT   ; Null start address.
0000    458
0000    459 ;
0000    460 ; Start command.
0000    461 ;
0000    462
0000    463          $STATE  STARTCMD                ; Command that starts executing
0000    464                                          ; code at a given address.
0000    465          $TRAN   'START'                 ; Command verb.
0000    466          $STATE
0000    467          $TRAN   !EXPLICIT_VALUE,-       ; Starting address.
0000    468                  TPA$_EXIT,SET_STARTADDR
0000    469
0000    470 ;
0000    471 ; Examine command.
0000    472 ;
0000    473
0000    474          $STATE  EXAMINECMD              ; Command that displays the
0000    475                                          ; contents of some location.
0000    476          $TRAN   'EXAMINE'               ; Command verb.
0000    477          $STATE
0000    478          $TRAN   !EXADEP_QUAL            ; Parse qualifiers
0000    479          $STATE
0000    480          $TRAN   !ANY_VALUE,TPA$_EXIT,-  ; Examine special location.
0000    481                  EXADEP_ADDR
0000    482
0000    483 ;
0000    484 ; Deposit command.
0000    485 ;
0000    486
0000    487          $STATE  DEPOSITCMD              ; Command to deposit data in
0000    488                                          ; a word or longword location.
```

```
                                        J 13
0000    489          $TRAN   'DEPOSIT'                ; Command verb.
0000    490          $STATE
0000    491          $TRAN   !EXADEP_QUAL             ; Parse qualifiers
0000    492          $STATE
0000    493          $TRAN   !EXPLICIT_VALUE,,-       ; Specify special location
0000    494                  EXADEP_ADDR
0000    495          $STATE
0000    496          $TRAN   !EXPLICIT_VALUE,-        ; Specify data to be deposited.
0000    497                  TPA$_EXIT,DEPOSIT_VALUE
0000    498
0000    499 ;
0000    500 ; Examine and deposit qualifiers.
0000    501 ;
0000    502
0000    503          $STATE  EXADEP_QUAL              ; Qualifiers.
0000    504          $TRAN   !LOC_QUAL                ; Location qualifiers
0000    505          $TRAN   TPA$_LAMBDA              ; Allow null location qualifier
0000    506          $STATE
0000    507          $TRAN   !SIZE_QUAL,TPA$_EXIT     ; Size qualifiers
0000    508
0000    509 ;
0000    510 ; Location qualifers.
0000    511 ;
0000    512          $STATE  LOC_QUAL
0000    513          $TRAN   '/'
0000    514          $TRAN   TPA$_LAMBDA,TPA$_EXIT
0000    515          $STATE
0000    516          $TRAN   'G',TPA$_EXIT,SET_GENERAL
0000    517          $TRAN   'I',TPA$_EXIT,SET_INTERNAL
0000    518          $TRAN   'P',TPA$_EXIT,SET_PHYSICAL
0000    519
0000    520 ;
0000    521 ; Size qualifiers
0000    522 ;
0000    523          $STATE  SIZE_QUAL
0000    524          $TRAN   '/'
0000    525          $TRAN   TPA$_LAMBDA,TPA$_EXIT
0000    526          $STATE
0000    527          $TRAN   'L',TPA$_EXIT,SET_LONG
0000    528          $TRAN   'W',TPA$_EXIT,SET_WORD
0000    529          $TRAN   'B',TPA$_EXIT,SET_BYTE
0000    530 ;
0000    531 ; Numeric value.
0000    532 ;
0000    533
0000    534          $STATE  NUMBER                   ; Accept a hexadecimal value.
0000    535          $TRAN   TPA$_HEX,TPA$_EXIT
0000    536          $TRAN   '@',TPA$_EXIT,-          ; Accept an at sign meaning
0000    537                  USE_LAST_VALUE           ; last value displayed.
0000    538
0000    539 ;
0000    540 ; Special location for an examine command.
0000    541 ;
0000    542
0000    543          $STATE  ANY_VALUE                ; Accept a special character
0000    544                                           ; indicating a location.
0000    545          $TRAN   !EXPLICIT_VALUE,-        ; Accept a deposit location.
```

```
0000    546                     TPA$_EXIT
0000    547             $TRAN   TPA$_LAMBDA,TPA$_EXIT,- ; Or accept a null examine
0000    548                     USE_NEXT_ADDR           ; location, meaning "next".
0000    549
0000    550 ;
0000    551 ; Special location for an examine or deposit command.
0000    552 ;
0000    553
0000    554             $STATE  EXPLICIT_VALUE          ; Accept "*", "+", or "P".
0000    555             $TRAN   !NUMBER,TPA$_EXIT       ; Accept any number.
0000    556             $TRAN   '*',TPA$_EXIT,-         ; Use location used in last
0000    557                     USE_LAST_ADDR           ; examine or deposit command.
0000    558             $TRAN   '+',TPA$_EXIT,-         ; Use next sequential location
0000    559                     USE_NEXT_ADDR           ; a word or longword from last
0000    560                                             ; location.
0000    561             $TRAN   '-',TPA$_EXIT,-         ; Use previous address
0000    562                     USE_PREVIOUS
0000    563
0000    564 ;
0000    565 ; Parse a file specification for the LOAD command and for indirect
0000    566 ; files.
0000    567 ;
0000    568
0000    569             $STATE  FILESPEC                ; Untangle a file specification.
0000    570             $TRAN   TPA$_STRING             ; Parse a file name.
0000    571             $STATE
0000    572             $TRAN   '.'                     ; Starts with a dot.
0000    573             $TRAN   TPA$_LAMBDA,TPA$_EXIT   ; Accept null file type.
0000    574             $STATE
0000    575             $TRAN   TPA$_STRING,TPA$_EXIT   ; Ends in a normal string.
0000    576             $TRAN   TPA$_LAMBDA,TPA$_EXIT   ; Can be null file type too.
0000    577
0000    578 ;
0000    579 ; Load qualifier.
0000    580 ;
0000    581
0000    582             $STATE  LOAD_QUAL               ; Process /START qualifier.
0000    583             $TRAN   '/'                     ; Starts with a slash.
0000    584             $STATE
0000    585             $TRAN   'START'                 ; Then the word 'START'.
0000    586             $STATE
0000    587             $TRAN   ':'                     ; Then a colon.
0000    588             $STATE
0000    589             $TRAN   !EXPLICIT_VALUE,-       ; Then a start address.
0000    590                     TPA$_EXIT,SET_STARTADDR
0000    591 ;
0000    592 ; Help command.
0000    593 ;
0000    594             $STATE  HELPCMD                 ; Print help text at console
0000    595             $TRAN   'HELP'                  ; Command verb
0000    596             $STATE
0000    597             $TRAN   TPA$_LAMBDA,TPA$_EXIT   ; No text expected after 'HELP'
0000    598
0000    599 ;
0000    600 ; Boot command.
0000    601 ;
0000    602             $STATE  BOOTCMD                 ; Boot from default boot file
```

```
0000    603          $TRAN   'BOOT'                      ; Command verb
0000    604          $STATE
0000    605          $TRAN   !DEVSPEC,TPA$_EXIT,SAV_DEVSPEC  ; Parse device spec
0000    606          $TRAN   TPA$_EOS,TPA$_EXIT,-        ; Accept null device spec
0000    607                  ,CMD_M_NULLDEV,COMMAND_FLAGS
0000    608
0000    609 ;
0000    610 ; Device spec.
0000    611 ;
0000    612          $STATE  DEVSPEC                     ; 3 character device spec
0000    613          $TRAN   !DEVICE_TYPE                ; Get the generic device type
0000    614          $STATE
0000    615          $TRAN   !CONTROLLER                 ; Get the controller designater
0000    616          $STATE
0000    617          $TRAN   TPA$_DECIMAL,,,,UNIT_NUMBER ; Get unit number
0000    618          $STATE
0000    619          $TRAN   ':',TPA$_EXIT               ; Parse optional colon
0000    620          $TRAN   TPA$_LAMBDA,TPA$_EXIT
0000    621
0000    622
0000    623 ;
0000    624 ; Device types.
0000    625 ;
0000    626          $STATE  DEVICE_TYPE                 ; Parse the 2 char generic device type
0000    627          $TRAN   TPA$_ALPHA                  ; Get first char
0000    628          $STATE
0000    629          $TRAN   TPA$_ALPHA,TPA$_EXIT        ; Get second char
0000    630
0000    631
0000    632 ;
0000    633 ; Parse the 1 char contoller designator.
0000    634 ;
0000    635          $STATE  CONTROLLER                  ; Parse a controller spec
0000    636          $TRAN   'A',TPA$_EXIT
0000    637          $TRAN   'B',TPA$_EXIT
0000    638
0000    639
0000    640
0000    641
0000    642 ;
0000    643 ; Indirect command files.
0000    644 ;
0000    645
0000    646          $STATE  CMDFILE                     ; Locate and execute a command
0000    647                                              ; file.
0000    648          $TRAN   '@'                         ; Starts with at sign.
0000    649          $STATE
0000    650          $TRAN   !FILESPEC,TPA$_EXIT,-       ; A file specification must
0000    651                  MOV_FILESPEC                ; follow.
0000    652
0000    653          $END_STATE                          ; End of syntax tables.
0000    654
0000    655 ;
0000    656 ; Declare a code PSECT that will always link at the start of the image.
0000    657 ;
0000    658
00000000    659          .PSECT  $$$$00BOOT,LONG
```

```
0000    660
0000    661         .SHOW   EXPANSIONS
0000    662
0000    663         .DEFAULT DISPLACEMENT, WORD
```

```
0000    665          .SBTTL  SETUP_BOOT58 - Set up boot environment
0000    666
0000    667 ;++
0000    668 ; Functional description:
0000    669 ;
0000    670 ;       When this program gains control, no SCB or RPB exists. This
0000    671 ;       routine identifies space for an SCB in the first free page
0000    672 ;       of memory following this program. The routine fills each
0000    673 ;       longword in the SCB with the address of a fault handler.
0000    674 ;       Then the routine loads the SCB address into the SCBB processor
0000    675 ;       register.
0000    676 ;
0000    677 ;       If the user requested XDELTA via a software boot control flag,
0000    678 ;       the routine then loads 2 SCB longwords with XDELTA handlers,
0000    679 ;       and creates the first entry in the XDELTA breakpoint table.
0000    680 ;
0000    681 ;       If the boot flag requested an initial breakpoint, the routine
0000    682 ;       then executes a BPT instruction, which passes control to
0000    683 ;       XDELTA.
0000    684 ;
0000    685 ;       This routine then calls BOOT58 -- the command processor -- to
0000    686 ;       accept and process commands from the console terminal.
0000    687 ;
0000    688 ; Inputs:
0000    689 ;
0000    690 ;       R5      - software boot control flags
0000    691 ;
0000    692 ;                 Bit     Meaning
0000    693 ;                 ---     -------
0000    694 ;
0000    695 ;                  5      RPB$V_BOOBPT.
0000    696 ;                         Bootstrap breakpoint. Stops BOOT58 in XDELTA
0000    697 ;                         before accepting commands.
0000    698 ;
0000    699 ;                 11      RPB$V_NODEFBOO.
0000    700 ;                         Do not automatically execute a BOOT command
0000    701 ;                         on  ctivation. (Default behavior is to a BOOT.)
0000    702 ;
0000    703 ;       R6      - address of 'evice ROM driver subroutine
0000    704 ;
0000    705 ;       SP      - <base_address + ^X200> of 64kb of good memory
0000    706 ;
0000    707 ; Implicit inputs:
0000    708 ;
0000    709 ;       BOOT_FAULT      - address of a fault handler
0000    710 ;       BOOTRIGH        - address of the first free byte beyond BOOT58
0000    711 ;                         code and data
0000    712 ;
0000    713 ;       When SET P_BOOT58 gains control, physical memory looks like
0000    714 ;       the diagram below:
0000    715 ;
0000    716 ;       SP-^X200:       +---------------------------------------+
0000    717 ;                       |              boot block 0             |
0000    718 ;       SP:             +---------------------------------------+
0000    719 ;                       |                                       |
0000    720 ;                       |                unused                 |
0000    721 ;                       |                                       |
```

```
                        ;       SP+^XBE00:      +-----------------------------------------+
                        ;                       |       Primary bootstrap (BOOT58)        |
                        ;       BOOTHIGH:       +-----------------------------------------+
                        ;                       |                                         |
                        ;                       |                 unused                  |
                        ;                       |                                         |
                        ;                       +-----------------------------------------+
                        ;
                        ; Implicit outputs:
                        ;
                        ;       DRIVER_SUBROUT  - address of the driver subroutine
                        ;       START_ADDRESS   - <base of good memory + ^X200>
                        ;       GOODMEM_BASE    - base address of 64K bytes of good memory
                        ;       FIRST_FREE_BYTE - address of first free byte past BOOT58
                        ;       PR$_SCBB        - address of an SCB filled with handler address
                        ;
                        ;       When SETUP_BOOT58 exits to the BOOT58 command processor,
                        ;       physical memory looks like the diagram below:
                        ;
                        ;       base+0:         +-----------------------------------------+
                        ;                       |              boot block 0               |
                        ;       base+^X200:     +-----------------------------------------+
                        ;                       |                                         |
                        ;                       |                 unused                  |
                        ;                       |                                         |
                        ;       base+^XC000:    +-----------------------------------------+
                        ;                       |                                         |
                        ;                       |                 BOOT58                  |
                        ;                       |                                         |
                        ;       BOOTHIGH:       +-----------------------------------------+
                        ;                       |                  stack                  |
                        ;       SP:             +-----------------------------------------+
                        ;
                        ;--
                        
00000001                .IF     NE,SAENVIRON            ; If operating standalone,
                                                        ; include this routine.

                        SETUP_BOOT58:                   ; Prepare a boot environment.

                        ;
                        ; Save the contents of R6. This is the address of the device ROM driver
                        ; subroutine. BOOT58 needs this later to read files from the TU58
                        ; device.
                        ;
                        
080E'CF    56    D0     MOVL    R6,DRIVER_SUBROUT       ; Save driver address.

                        ;
                        ; The BOOT58 program is loaded at ^XC000 bytes past the start of good
                        ; memory. This permits VMB or other primary bootstraps to be loaded at
                        ; the familiar ^X200 offset. Adjust SP for now to create SCB and stack.
                        ; Save address of the base of good memory, and default start address.
                        ;
                        
0048'CF    5E    D0     MOVL    SP,START_ADDRESS        ; Save original SP as user's
                                                        ; default load address.
```

```
0024'CF    FE00 CE   9E  000A   779          MOVAB   -^X200(SP),GOODMEM_BASE ; Save base of good memory too.
      0000BE00 8F   C0  0011   780          ADDL2   #<BOOT58_OFFSET-^X200>,-; Offset SP to address start of
               5E       0017   781                  SP                      ; BOOT58 code.
                        0018   782
                        0018   783 ;
                        0018   784 ; Reserve space for a System Control Block (SCB) immediately after the
                        0018   785 ; BOOT58 code. Write the address of a machine check fault handler in
                        0018   786 ; all vectors in the SCB. This handler is used during bootstrapping
                        0018   787 ; except when the bootstrap code specifically writes a different
                        0018   788 ; handler address in one of the SCB vectors.
                        0018   789 ;
                        0018   790 ; The low bit set in the address of the fault handler causes the
                        0018   791 ; handler to execute on the interrupt stack.
                        0018   792 ;
                        0018   793
          56   088D'CF   9E  0018   794          MOVAB   BOOT_FAULT+1,R6         ; Get the address of a handler.
          57   0200'CF   9E  001D   795          MOVAB   BOOTRIGH+^X200,R7       ; Get the address of the first
                          0022   796                                          ; byte past the SCB.
                          0022   797
                          0022   798 FILL_SCB:                                 ; Fill the SCB.
               77   56   D0  0022   799          MOVL    R6,-(R7)                ; Write 1 vector.
          57   01FF 8F   B3  0025   800          BITW    #^X1FF,R7               ; Check for page boundary.
                    F6   12  002A   801          BNEQ    FILL_SCB                ; Not yet. Write another vector.
               11   57   DA  002C   802          MTPR    R7,#PR$_SCBB            ; SCB full. Write SCB address
                         002F   803                                          ; into processor register.
                         002F   804
                         002F   805 ;
                         002F   806 ; Create a stack, and save the address of the first free byte past the
                         002F   807 ; space used by BOOT58 and its supporting environment.
                         002F   808 ;
                         002F   809
          5E   0800 C7   9E  002F   810          MOVAB   <4*^X200>(R7),SP        ; Create a 3 page stack.
          0028'CF   5E   D0  0034   811          MOVL    SP,FIRST_FREE_BYTE      ; Save address of 1st byte of
                         0039   812                                          ; free physical memory.
                         0039   813
                         0039   814 ;
                         0039   815 ; If XDELTA is linked into this program, set up 2 XDELTA handlers in
                         0039   816 ; the SCB -- for breakpoint and trace trap. Then initialize the XDELTA
                         0039   817 ; breakpoint table, and, if requested, execute a breakpoint before
                         0039   818 ; proceeding with the bootstrap.
                         0039   819 ;
                         0039   820
                00000001 0039   821          .IF     NE,SADEBUG              ; If XDELTA is linked, do next.
     2C A7    0000'CF   9E  0039   822          MOVAB   XDELBPT,^X2C(R7)        ; Plug BPT handler.
     28 A7    0000'CF   9E  003F   823          MOVAB   XDELTBIT,^X28(R7)       ; Plug trace trap handler.
0000'CF       0050'CF   9E  0045   824          MOVAB   INI$BRK,XDELIBRK        ; Store initial breakpoint.
        01 55    05   E1  004C   825          BBC     #RPB$V_BOOBPT,R5,-      ; If no breakpoint is wanted,
                         0050   826                  NOBRK                   ; just continue in BOOT58.
                         0050   827
                         0050   828 ;
                         0050   829 ; Initial breakpoint. Registers usage is as follows:
                         0050   830 ;
                         0050   831 ;       R0-R5   - unchanged
                         0050   832 ;       R6      - address (+1) of a bootstrap fault handler
                         0050   833 ;       R7      - address of the SCB
                         0050   834 ;       R8-R11  - unchanged
                         0050   835 ;       AP-FP   - unchanged
```

```
                     0050   836 ;       SP      - address of a 3-page stack
                     0050   837 ;
                     0050   838
                     0050   839 INI$BRK::                                ; Breakpoint
                  03 0050   840         BPT                              ; Stop in XDELTA.
                     0051   841
                     0051   842 NOBRK:                                   ; Proceed with BOOT58.
                     0051   843         .ENDC                            ; End of XDELTA conditional.
                     0051   844
                     0051   845 ;
                     0051   846 ; The environment is okay. Now call the command processor.
                     0051   847 ;
                     0051   848
0057'CF    00     FB 0051   849         CALLS   #0,BOOT58                ; Accept and process commands.
                  00 0056   850         HALT                             ; For now, just halt on return.
                     0057   851
                     0057   852         .ENDC                            ; End of standalone conditional.
```

```
                        0057   854          .SBTTL  BOOT58 - Process bootstrap commands
                        0057   855
                        0057   856 ;++
                        0057   857 ; Functional description:
                        0057   858 ;
                        0057   859 ;       The command processor saves current register values, sets a
                        0057   860 ;       command options longword to a default state, and then accepts
                        0057   861 ;       commands until a START command transfers control to some other
                        0057   862 ;       program, or the user types <control-P> to return control to the
                        0057   863 ;       CONSOLE program.
                        0057   864 ;
                        0057   865 ; Implicit inputs:
                        0057   866 ;
                        0057   867 ;       COMMAND_FLAGS   - a location to hold the default or modified
                        0057   868 ;                         command options.
                        0057   869 ;       PARAM_BLOCK     - address of the TPARSE parameter block. Fields
                        0057   870 ;                         of interest are the following:
                        0057   871 ;
                        0057   872 ;               TPA$L_COUNT     - # of longwords in block
                        0057   873 ;               TPA$L_OPTIONS   - TPARSE flag bits
                        0057   874 ;
                        0057   875 ;                                 TPA$M_BLANKS - treat blanks explicitly
                        0057   876 ;                                                instead of invisibly
                        0057   877 ;                                 TPA$M_ABBREV - permit abbreviations
                        0057   878 ;
                        0057   879 ;               TPA$L_STRINGCNT - # characters remaining in input string
                        0057   880 ;               TPA$L_STRINGPTR - address of remainder of input string
                        0057   881 ;               TPA$L_TOKENCNT  - # characters in current token
                        0057   882 ;               TPA$L_TOKENPTR  - address of current token
                        0057   883 ;               TPA$L_CHAR      - character matched by a single
                        0057   884 ;                                 character symbol type
                        0057   885 ;               TPA$L_NUMBER    - binary value of a numeric token
                        0057   886 ;               TPA$L_PARAM     - 32-bit parameter from state transition
                        0057   887 ;
                        0057   888 ;       TPA$K_LENGTH0   - # bytes in TPARSE parameter block
                        0057   889 ;       COMMAND_STRING  - buffer for input string
                        0057   890 ;       COMMAND_BUFFER  - buffer for a single line of input
                        0057   891 ;       CMD_BUFFER_SIZE - length of buffer for single line of input
                        0057   892 ;       BOO$GQ_CMDESC   - string descriptor for RMS input
                        0057   893 ;       KEY_TABLE       - address of TPARSE keyword table
                        0057   894 ;       BOOT58_TABLE    - address of TPARSE state table
                        0057   895 ;
                        0057   896 ; Outputs:
                        0057   897 ;
                        0057   898 ;       R0-FP   - original or user-modified values
                        0057   899 ;
                        0057   900 ;--
                        0057   901
                  0FFC  0057   902          .ENTRY  BOOT58,-
                        0059   903                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0059   904
                        0059   905 ;
                        0059   906 ; Save R1-PC on the stack. Store the stack address
                        0059   907 ; of saved R0 in own storage.
                        0059   908 ;
                        0059   909
0030'CF    5E    D0     0059   910          MOVL    SP,CURRENT_ADDRESS      ; Save SP in own storage.
```

```
                00   DD  005E   911          PUSHL   #0                              ; Then push a 0 to represent the
          0030'CF    DD  0060   912          PUSHL   CURRENT_ADDRESS                 ; original PC, and push SP.
             3FFF 8F BB  0064   913          PUSHR   #^M<R0,R1,R2,R3,R4,R5,-         ; Save all the other registers
                     0068   914                  R6,R7,R8,R9,R10,R11,-           ; on the stack too.
                     0068   915                  AP,FP>
    0040'CF    5E   D0  0068   916          MOVL    SP,REGISTER_TABLE               ; Save address of saved register
                     006D   917                                                  ; values.
                     006D   918
                     006D   919 ;
                     006D   920 ; Initialize the mask that describes default address interpretation
                     006D   921 ; and value display mode.
                     006D   922 ;
                     006D   923
          002C'CF    D4  006D   924          CLRL    COMMAND_FLAGS                   ; Initialize command settings.
      00000100 8F    C8  0071   925          BISL    #CMD_M_PHYSICAL,-               ; Set to interpret addresses
          002C'CF        0077   926                  COMMAND_FLAGS                   ; as physical.
    080A'CF    5E   D0  007A   927          MOVL    SP,L_OLD_STACKP                 ; Save current stack pointer to
                     007F   928                                                  ; allow restoration on
                     007F   929                                                  ; exceptions.
    0809'CF    01   90  007F   930          MOVB    #1,B_ENVIRON_SAFE               ; Note that environment is
                     0084   931                                                  ; initialized.
                     0084   932
          0054'CF    DE  0084   933          MOVAL   HELP_FILE_SPEC,-                ; Set up help fie descripter
          0062'CF        0088   934                  HELP_FILE_DESCRIP+4             ;
          0066'CF    DE  008B   935          MOVAL   DEFBOO_SPEC,-                   ; Set up defalut boot command file spec
          0074'CF        008F   936                  DEFBOO_DESCRIP+4                ;
          0078'CF    DE  0092   937          MOVAL   BOOTFILE_SPEC,-                 ; Set up optional boot command file spec
          0086'CF        0096   938                  BOOTFILE_DESCRIP+4              ;
                     0099   939
                     0099   940 READ_COMMAND:                                       ; Main command processing loop.
57  0000'CF    9E   0099   941          MOVAB   PARAM_BLOCK,R7                      ; Get parameter block address.
                     009E   942
                     009E   943 ;
                     009E   944 ; Initialize string buffer address and length in the TPARSE parameter
                     009E   945 ; block. Then transfer a 1 to n line command into the string buffer.
                     009E   946 ;
                     009E   947
            08 A7    D4  009E   948          CLRL    TPA$L_STRINGCNT(R7)             ; Initialize string length.
          01A3'CF    9E  00A1   949          MOVAB   COMMAND_STRING,-                ; Initialize string buffer
            0C A7        00A5   950                  TPA$L_STRINGPTR(R7)             ; address.
                     00A7   951
                     00A7   952 ;
                     00A7   953 ; Check for non-interactive input: if taking commands from an indirect
                     00A7   954 ; file, the file is in memory. Call a routine to get the next record.
                     00A7   955 ;
                     00A7   956
                     00A7   957 READ_A_LINE                                      ; Process one line.
              07     E1  00A7   958          BBC     #CMD_V_INDIRECT,-               ; If processing interactive
31 002C'CF           00A9   959                  COMMAND_FLAGS,10$               ; commands, branch.
          01BD'CF    16  00AD   960          JSB     GET_A_RECORD                    ; For indirect, get a record.
            53 50    E9  00B1   961          BLBC    R0,20$                          ; Branch if error
                     00B4   962 ;
                     00B4   963 ; Put out a <cr><lf>.
                     00B4   964 ;
              7E     7C  00B4   965          CLRQ    -(SP)                           ; Create a null descrptor
          07B8'CF    9F  00B6   966          PUSHAB  LINE_TERM                       ; Address of <cr><lf>
0000'CF     00    FB  00BA   967          CALLS   #0,BOO$READPROMPT               ; "Print" the <cr><lf>
```

```
                45 50   E9  00BF   968          BLBC    R0,20$                  ; Branch if error
                            00C2   969 ;
                            00C2   970 ; Echo the command line to the console.  A special ASCIZ string is put
                            00C2   971 ; on the end of the buffer to cause a <cr><lf> after printing the line.
                            00C2   972 ;
            51   62   9A    00C2   973          MOVZBL  (R2),R1                 ; Restore R1 (overwritten by BOO$READPROMPT)
    54   00DC'CF   9E    00C5   974          MOVAB   COMMAND_BUFFER+1,R4     ; Get address of first char
    50   54   51   C1    00CA   975          ADDL3   R1,R4,R0                ; Calculate address of last char
    60   07B8'CF   D0    00CE   976          MOVL    LINE_TERM,(R0)          ; Put line term past last char
                7E   7C    00D3   977          CLRQ    -(SP)                   ; Create null descripter
                54   DD    00D5   978          PUSHL   R4                      ; Push address of first char
 0000'CF        03   FB    00D7   979          CALLS   #3,BOO$READPROMPT       ; Echo command line
                29   11    00DC   980          BRB     20$                     ; Go check for success.
                            00DE   981
                            00DE   982 ;
                            00DE   983 ; If this is the first time thru and the 'disable default boot' flag is
                            00DE   984 ; not set, then execute a 'BOOT' command to invoke the DEFBOO command
                            00DE   985 ; file on the console TU58.  This action mimics that of the 11-780 on
                            00DE   986 ; reboot if the AUTO-RESTART switch is ON.
                            00DE   987 ;
                            00DE   988 ; Interactive input from console terminal:
                            00DE   989 ;
                            00DE   990 ; Load an input buffer address, length in bytes, and address of a prompt
                            00DE   991 ; string onto the stack. Then call a subroutine to prompt for and read
                            00DE   992 ; an input line from the terminal. If the read fails, return to the
                            00DE   993 ; caller with an error status code.
                            00DE   994 ;
                            00DE   995
                            00DE   996 10$:                                     ; Interactive input.
52   00DB'CF   9E    00DE   997          MOVAB   COMMAND_BUFFER,R2       ; Get address of command buffer.
     50   05   D0    00E3   998          MOVL    #SAVED_R5,R0            ; Index for R5
 00000800 8F   D3    00E6   999          BITL    #RPB$M_NODEFBOO,-       ; Skip possible unattend boot if
   0040'DF40           00EC  1000           @REGISTER_TABLE[R0]     ;  flag set in R5
            06   12    00F0  1001          BNEQ    15$
            0F   E3    00F2  1002          BBCS    #CMD_V_UNATTEND,-       ; If unattended, use the BOOT command
     002C'CF           00F4  1003           COMMAND_FLAGS,-         ;  built into the command buffer
            13         00F7  1004           GOT_A_LINE
                       00F8  1005 15$:
            52   DD    00F8  1006          PUSHL   R2                      ; Push address on stack.
  7E    C4   8F   9A    00FA  1007          MOVZBL  #CMD_BUFFER_SIZE,-(SP)  ; Push buffer size on stack.
     07AE'CF   9F    00FE  1008          PUSHAB  PROMPT_STRING           ; Push prompt address on stack.
                       0102  1009
                       0102  1010 ;
                       0102  1011 ; If debugging with DEBUG in a timesharing environment, set up an
                       0102  1012 ; RMS input descriptor for BOO$READPROMPT.
                       0102  1013 ;
                       0102  1014
           00000000    0102  1015          .IF     NE,TSDEBUG
                       0102  1016          CLRL    BOO$GQ_CMDESC           ; Zero the # chars in buffer.
                       0102  1017          .ENDC
                       0102  1018
0000'CF    03   FB    0102  1019          CALLS   #3,BOO$READPROMPT       ; Prompt for and read an input
                       0107  1020                                          ; line.
                       0107  1021
                       0107  1022 20$:                                     ; Check for error on input.
      01 50   E8    0107  1023          BLBS    R0,GOT_A_LINE           ; Branch on success.
                04   010A  1024          RET                             ; Return to caller on failure.
```

```
                     010B  1025
                     010B  1026 ;
                     010B  1027 ; The command buffer holds an input line. Calculate the 1st free byte
                     010B  1028 ; in the output buffer. One character at a time, copy the characters
                     010B  1029 ; in the input buffer into the output buffer.
                     010B  1030 ;
                     010B  1031
                     010B  1032 GOT_A_LINE:                                  ; Process an input line.
      0C A7    C1    010B  1033         ADDL3   TPA$L_STRINGPTR(R7),-       ; Add start of output buffer to
53    08 A7          010E  1034                 TPA$L_STRINGCNT(R7),R3      ; to count of characters in that
                     0111  1035                                             ; buffer to point to free byte.
   50    82    9A    0111  1036         MOVZBL  (R2)+,R0                    ; Get length of input line.
         91    13    0114  1037         BEQL    READ_A_LINE                 ; If null line, get another.
   51    52    D0    0116  1038         MOVL    R2,R1                       ; Need input buffer address in
                     0119  1039                                             ; R1 for a LOCC instruction.
                     0119  1040
                     0119  1041 ;
                     0119  1042 ; Pick up one character at a time. If a hyphen, this may be a
                     0119  1043 ; continuation line; branch to special handler. If an exclamation
                     0119  1044 ; point, this is a comment; skip characters until end of comment.
                     0119  1045 ; Otherwise, increment the character count, store the characters in
                     0119  1046 ; the output buffer, decrement the count of input characters left,
                     0119  1047 ; and, if count is nonzero, loop through here again.
                     0119  1048 ;
                     0119  1049
                     0119  1050 GET_A_CHARACTER:                            ; Loop to read characters.
   52    81    9A    0119  1051         MOVZBL  (R1)+,R2                    ; Get a character.
                     011C  1052 ;       CMPB    #^A/-/,R2                   ; Might this be continuation?
                     011C  1053 ;       BEQL    HYPHEN_IN_LINE              ; Yes. Go process it.
   52    21    91    011C  1054         CMPB    #^A/!/,R2                   ; Is this a comment?
         0B    13    011F  1055         BEQL    BYPASS_COMMENT              ; Yes. Go bypass it.
                     0121  1056
                     0121  1057 ;
                     0121  1058 ; Normal character. Increment count in TPARSE parameter block, and
                     0121  1059 ; write character to output buffer. Then loop for next character
                     0121  1060 ; unless input line exhausted.
                     0121  1061 ;
                     0121  1062
   08 A7    D6    0121  1063         INCL    TPA$L_STRINGCNT(R7)         ; Increment character count.
83   52    90    0124  1064         MOVB    R2,(R3)+                    ; Write output character.
                     0127  1065
                     0127  1066 MORE_CHARS:                                 ; See if more characters.
   EF 50    F5    0127  1067         SOBGTR  R0,GET_A_CHARACTER          ; Decrement character count,
                     012A  1068                                             ; branch if line not finished.
      50    11    012A  1069         BRB     PARSE_COMMAND               ; Command all collected. Go
                     012C  1070                                             ; parse the command.
                     012C  1071
                     012C  1072 ;
                     012C  1073 ; Comment character: skip all characters until another exclamation
                     012C  1074 ; point or end of line. A second exclamation point ends the comment,
                     012C  1075 ; and normal character processing can resume. End of line indicates
                     012C  1076 ; that the entire command is in, and the line can be parsed. Note that
                     012C  1077 ; the LOCC instruction updates R1 and R0 appropriately.
                     012C  1078 ;
                     012C  1079
                     012C  1080 BYPASS_COMMENT:                             ; Skip a comment.
      50    D7    012C  1081         DECL    R0                          ; Decrement character count
```

```
                     012E  1082                                       ; for the comment mark.
01 A1   50   21   3A 012E  1083          LOCC    #^A/!/,R0,1(R1)      ; Look for another "!".
             47   13 0133  1084          BEQL    PARSE_COMMAND        ; None exists. Go parse command.
             51   D6 0135  1085          INCL    R1                   ; Found one. Move to next char.
             EE   11 0137  1086          BRB     MORE_CHARS           ; Continue getting characters.
                     0139  1087
                     0139  1088 ;
                     0139  1089 ; Line continuation character: save current output buffer pointer and
                     0139  1090 ; length for now. Scan characters after the hyphen. If all blanks until
                     0139  1091 ; a comment character or end of line, this is really a continuation
                     0139  1092 ; character. Otherwise, this is just a hyphen (with some other syntactic
                     0139  1093 ; value).
                     0139  1094 ;
                     0139  1095
                     0139  1096 HYPHEN_IN_LINE:                       ; See why a hyphen is here.
    54   53   D0      0139  1097          MOVL    R3,R4                ; Save output buffer pointer.
55    08   A7   D0    013C  1098          MOVL    TPA$L_STRINGCNT(R7),R5 ; Save output char. count.
  83   52   90        0140  1099          MOVB    R2,(R3)+             ; Write hyphen into output
                     0143  1100                                       ; string.
    08   A7   D6      0143  1101          INCL    TPA$L_STRINGCNT(R7)  ; And count the character.
         1A   11      0146  1102          BRB     NEXT_CHAR            ; See if more characters.
                     0148  1103
                     0148  1104 ;
                     0148  1105 ; Another character in line. If blank, keep looking. If comment, skip
                     0148  1106 ; over the comment, and then keep looking. If anything but a blank,
                     0148  1107 ; assume the hyphen was NOT a line continuation, and resume normal
                     0148  1108 ; character processing.
                     0148  1109 ;
                     0148  1110
                     0148  1111 LOOK_FOR_BLANK:                       ; Check for blank or "!".
52   81   9A          0148  1112          MOVZBL  (R1)+,R2             ; Get next character.
83   52   90          014B  1113          MOVB    R2,(R3)+             ; Store in output string.
  08   A7   D6        014E  1114          INCL    TPA$L_STRINGCNT(R7)  ; Increment output count.
52   20   91          0151  1115          CMPB    #^A/ /,R2            ; Is character a blank?
       0C   13        0154  1116          BEQL    NEXT_CHAR            ; Yes. Move to next char.
52   09   91          0156  1117          CMPB    #^A/    /,R2         ; Is character a tab?
       07   13        0159  1118          BEQL    NEXT_CHAR            ; Yes. Move to next char.
52   21   91          015B  1119          CMPB    #^A/!/,R2            ; Is character an "!"?
       0F   13        015E  1120          BEQL    SKIP_COMMENT         ; Yes. Move past the comment.
       C5   11        0160  1121          BRB     MORE_CHARS           ; No. This is not a line
                     0162  1122                                       ; continuation.
                     0162  1123
                     0162  1124 ;
                     0162  1125 ; Keep hunting for blank, non-blank, exclamation point, or end of line.
                     0162  1126 ; If character count is exhausted, fall through.
                     0162  1127 ;
                     0162  1128
                     0162  1129 NEXT_CHAR:                           ; See if more characters.
  E3 50   F5         0162  1130          SOBGTR  R0,LOOK_FOR_BLANK    ; If a character, look at it.
                     0165  1131
                     0165  1132 ;
                     0165  1133 ; Found end of line. Restore the TPARSE parameter block command buffer
                     0165  1134 ; pointer and command character count to their state before the hyphen
                     0165  1135 ; was found. Then go on to read another line.
                     0165  1136 ;
                     0165  1137
                     0165  1138 END_OF_LINE:                         ; Found end of line.
```

```
                53   54    D0  0165  1139          MOVL    R4,R3                   ; Restore saved pointer to
          08 A7   55    D0  0168  1140          MOVL    R5,TPA$L_STRINGCNT(R7)  ; command string and count.
              FF38    31  016C  1141          BRW     READ_A_LINE             ; Read another line.
                          016F  1142
                          016F  1143 ;
                          016F  1144 ; After a hyphen, found a comment delimiter. Search for a comment
                          016F  1145 ; ending delimiter. If find one, go on looking for a blank or non-blank
                          016F  1146 ; to interpret the hyphen. If end of line, just restore command
                          016F  1147 ; pointers and read next line.
                          016F  1148 ;
                          016F  1149
                          016F  1150 SKIP_COMMENT:                            ; Skip command after hyphen.
                50    D7  016F  1151          DECL    R0                      ; Ignore the exclamation point.
    01 A1       50    21  3A  0171  1152          LOCC    #^A/!/,R0,1(R1)         ; Is there another "!"?
                ED    13  0176  1153          BEQL    END_OF_LINE             ; No. End of line. Go to next.
                51    D6  0178  1154          INCL    R1                      ; Yes. Move past it.
                E6    11  017A  1155          BRB     NEXT_CHAR               ; And keep looking for blank.
                          017C  1156
                          017C  1157 ;
                          017C  1158 ; Parse: initialize parsing state. Push the address of the keyword
                          017C  1159 ; table, the state table, and the TPARSE parameter block onto the
                          017C  1160 ; stack, and call TPARSE.
                          017C  1161 ;
                          017C  1162
                          017C  1163 PARSE_COMMAND:                           ; Command is all here. Parse it.
                08    D0  017C  1164          MOVL    #TPA$K_COUNTO,-         ; ???
                67        017E  1165                  TPA$L_COUNT(R7)
                02    C8  017F  1166          BISL    #TPA$M_ABBREV,-         ; Permit abbreviations in the
              04 A7       0181  1167                  TPA$L_OPTIONS(R7)       ; command keywords.
          0000'CF   9F  0183  1168          PUSHAB  KEY_TABLE               ; Pass address of keyword table,
          0000'CF   9F  0187  1169          PUSHAB  BOOT58_TABLE            ; state table, and
                57    DD  018B  1170          PUSHL   R7                      ; TPARSE parameter block.
      0000'CF   03    FB  018D  1171          CALLS   #3,LIB$TPARSE           ; Parse the command.
          0808'CF   E8  0192  1172          BLBS    B_ERROR_OUT,-           ; If error message already
                16        0196  1173                  CLEAR_ERRBIT            ; output, get next command.
             17 50    E8  0197  1174          BLBS    R0,NEXT_COMMAND         ; If success, get another.
                          019A  1175          MSG     <-E-Syntax error>       ; Output error message.
               00B6   30  019A                BSBW    BOO$FACMSG
72 65 20 78 61 74 6E 79 53 2D 45 2D  019D                .ASCIZ  \-E-Syntax error\
                00 72 6F 72  01A9
                          01AD
                          01AD  1176
                          01AD  1177 CLEAR_ERRBIT:                            ; Clear flag.
          0808'CF   94    01AD  1178          CLRB    B_ERROR_OUT             ; Clear error output flag.
                          01B1  1179
                          01B1  1180 ;
                          01B1  1181 ; Branch back through the command loop.
                          01B1  1182 ;
                          01B1  1183
                          01B1  1184 NEXT_COMMAND:                            ; Process another command.
                          01B1  1185 ;
                          01B1  1186 ; Make sure the command flag was cleared.  The INSV instruction
                          01B1  1187 ; will clear all the bits in COMMAND_FLAGS that indicate that a
                          01B1  1188 ; command is awaiting execution.
                          01B1  1189 ;
                50    D4  01B1  1190          CLRL    R0                      ; Generate 32 bits of 0's
                50    F0  01B3  1191          INSV    R0,-                    ; Source
```

```
                     01B5  1192                    #CMD_V_LOADCMD,-        ; Starting position
07   00              01B5  1193                    #NUM_OF_CMDS,-          ; Length
002C'CF              01B7  1194                    COMMAND_FLAGS           ; Destination
   FEDC   31         01BA  1195          BRW       READ_COMMAND
```

```
                        01BD  1197          .SBTTL  GET_A_RECORD - Extracts an RT-11 format record
                        01BD  1198
                        01BD  1199 ;++
                        01BD  1200 ; Functional description:
                        01BD  1201 ;
                        01BD  1202 ;       One block of the command file is in memory. A pointer indicates
                        01BD  1203 ;       the next byte to be read. This routine collects a record by
                        01BD  1204 ;       reading a character a time from memory. If end of block is
                        01BD  1205 ;       reached, this routine reads in the next block (if any).
                        01BD  1206 ;
                        01BD  1207 ;       At end of file, this routine turns off the indirect command
                        01BD  1208 ;       file processing flag, and returns a success status code. The
                        01BD  1209 ;       outer parser will process the null command and then prompt
                        01BD  1210 ;       for further interactive input.
                        01BD  1211 ;
                        01BD  1212 ;       RT-11 test files are in ASCII stream format. The format is as
                        01BD  1213 ;       follows:
                        01BD  1214 ;
                        01BD  1215 ;               no character count associated with the record
                        01BD  1216 ;               the record begins with the ASCII text
                        01BD  1217 ;               a record ends in a FF, LF, or VT
                        01BD  1218 ;               bytes containing CR or null are ignored
                        01BD  1219 ;
                        01BD  1220 ; Implicit inputs:
                        01BD  1221 ;
                        01BD  1222 ;       IND_CMDBUF_PTR  - contains address of next byte to read
                        01BD  1223 ;       IND_CMDBUFFER   - starting buffer address of command file buffer
                        01BD  1224 ;       512             - number of bytes in command file buffer
                        01BD  1225 ;       IND_W_NEXT_LBN  - next LBN in the command file
                        01BD  1226 ;       IND_W_BLKS_LEFT - blocks left to read from the command file
                        01BD  1227 ;
                        01BD  1228 ; Outputs:
                        01BD  1229 ;
                        01BD  1230 ;       This routine preserves registers R5-R11.
                        01BD  1231 ;
                        01BD  1232 ; Implicit outputs:
                        01BD  1233 ;
                        01BD  1234 ;       COMMAND_BUFFER  - byte 0 is the record size in bytes
                        01BD  1235 ;                         bytes 1-n are the ASCII text of the record
                        01BD  1236 ;                         buffer contains no FFs, LFs, CRs, or VTs
                        01BD  1237 ;
                        01BD  1238 ;       The static variables that point to the indirect command file
                        01BD  1239 ;       and its data are updated. These variables are:
                        01BD  1240 ;
                        01BD  1241 ;               IND_CMDBUF_PTR
                        01BD  1242 ;               IND_CMDBUFFER
                        01BD  1243 ;               IND_W_NEXT_LBN
                        01BD  1244 ;               IND_W_BLKS_LEFT
                        01BD  1245 ;
                        01BD  1246 ;--
                        01BD  1247
                        01BD  1248 GET_A_RECORD:                            ; Collect 1 record.
          51    D4      01BD  1249          CLRL    R1                      ; Clear character count.
53  07A3'CF     D0      01BF  1250          MOVL    IND_CMDBUF_PTR,R3       ; Get source address.
52  00DC'CF     9E      01C4  1251          MOVAB   COMMAND_BUFFER+1,R2     ; Get destination address.
54  07A3'CF     9E      01C9  1252          MOVAB   IND_CMDBUFFER+512,R4    ; Get address of 1 byte past end
                        01CE  1253                                          ; of buffer.
```

```
                                01CE  1254
                                01CE  1255 GET_CHAR:                                    ; Read 1 character.
                          54    53   D1   01CE  1256          CMPL    R3,R4            ; Are we at the end of buffer?
                                47   1F   01D1  1257          BLSSU   20$              ; No. Branch to get character.
                          07A7'CF    B5   01D3  1258          TSTW    IND_W_BLKS_LEFT  ; Any blocks left to read?
                                61   15   01D7  1259          BLEQ    END_OF_FILE      ; No. Process end of file.
                                06   BB   01D9  1260          PUSHR   #^M<R1,R2>       ; Save R1 and R2
                          05A3'CF    9F   01DB  1261          PUSHAB  IND_CMDBUFFER    ; Send address of buffer for a
                          07A9'CF    DD   01DF  1262          PUSHL   IND_W_NEXT_LBN   ; block, and an LBN.
                          07A9'CF    B6   01E3  1263          INCW    IND_W_NEXT_LBN   ; Increment next block number.
                          07A7'CF    B7   01E7  1264          DECW    IND_W_BLKS_LEFT  ; Decrement blocks left.
                    0000'CF    02   FB   01EB  1265          CALLS   #2,FIL$READ_LBN  ; Read a block.
                                06   BA   01F0  1266          POPR    #^M<R1,R2>       ; Restore R1 and R2
                              19 50   E8   01F2  1267          BLBS    R0,10$           ; Branch on success.
                                01F5  1268          MSG     <-E-Cannot read file>
                                 005B   30   01F5          BSBW    BOO$FACMSG
65 72 20 74 6F 6E 6E 61 43 2D 45 2D   01F8          .ASCIZ  \-E-Cannot read file\
           00 65 6C 69 66 20 64 61    0204
                                020C
                                44   11   020C  1269          BRB     EXIT_GETRECORD   ; Return with the error.
                                020E  1270
                                020E  1271 10$:                                         ; New block of file in memory.
                          05A3'CF    DE   020E  1272          MOVAL   IND_CMDBUFFER,-  ; Set up buffer pointer.
                          07A3'CF         0212  1273                  IND_CMDBUF_PTR
                    53    07A3'CF    D0   0215  1274          MOVL    IND_CMDBUF_PTR,R3 ; Set up in register too.
                                021A  1275
                                021A  1276 ;
                                021A  1277 ; Register usage below is as follows:
                                021A  1278 ;
                                021A  1279 ;       R0      - holds a character
                                021A  1280 ;       R1      - count of characters collected so far
                                021A  1281 ;       R2      - address of next free byte in output buffer
                                021A  1282 ;       R3      - address of next byte to read
                                021A  1283 ;
                                021A  1284
                                021A  1285 20$:                                         ; Buffer is okay. Get a char.
                          50    83   9A   021A  1286          MOVZBL  (R3)+,R0         ; Get a character.
                                AF   13   021D  1287          BEQL    GET_CHAR         ; Skip nulls.
                          0D    50   91   021F  1288          CMPB    R0,#CAR_RETURN   ; Carriage return?
                                AA   13   0222  1289          BEQL    GET_CHAR         ; Skip them too.
                          0A    50   91   0224  1290          CMPB    R0,#LINE_FEED    ; Line feed?
                                17   13   0227  1291          BEQL    END_OF_RECORD    ; Yes. Branch.
                          0B    50   91   0229  1292          CMPB    R0,#VERTICAL_TAB ; Vertical tab?
                                12   13   022C  1293          BEQL    END_OF_RECORD    ; Yes. Branch.
                          0C    50   91   022E  1294          CMPB    R0,#FORM_FEED    ; Form feed?
                                0D   13   0231  1295          BEQL    END_OF_RECORD    ; Yes. Branch.
                                51   D6   0233  1296          INCL    R1               ; Increment character count.
                          82    50   90   0235  1297          MOVB    R0,(R2)+         ; Save the character.
                                94   11   0238  1298          BRB     GET_CHAR         ; Get the next character.
                                023A  1299
                                023A  1300 ;
                                023A  1301 ; End of file has been reached. Turn off the indirect command file
                                023A  1302 ; processing flag, and return with success to caller.
                                023A  1303 ;
                                023A  1304
                                023A  1305 END_OF_FILE:                                 ; End of command file.
                            80  8F   8A   023A  1306          BICB    #CMD_M_INDIRECT,- ; Clear command file flag in
```

```
                002C'CF        023D  1307                    COMMAND_FLAGS           ; command flag longword.
                               0240  1308
                               0240  1309 END_OF_RECORD:                            ; End of record seen.
00DB'CF   51    90             0240  1310          MOVB      R1,COMMAND_BUFFER       ; Store record count.
52   00DB'CF    9E             0245  1311          MOVAB     COMMAND_BUFFER,R2       ; Restore output buffer address
                               024A  1312                                            ; in a register.
07A3'CF   53    D0             024A  1313          MOVL      R3,IND_CMDBUF_PTR       ; Update buffer pointer.
                               024F  1314
                               024F  1315 RETURN_RECORD:                            ; Successful record collection.
                50    01       3C    024F  1316          MOVZWL    #SS$_NORMAL,R0          ; Set success status code.
                               0252  1317
                               0252  1318 EXIT_GETRECORD:                           ; Restore registers and return.
                               05    0252  1319          RSB                       ; Return.
```

```
                                0253  1321          .SBTTL  Output message routines
                                0253  1322
                                0253  1323 ;++
                                0253  1324 ; BOO$FACMSG    - Output facility-soecific error message
                                0253  1325 ;
                                0253  1326 ; Functional description:
                                0253  1327 ;
                                0253  1328 ;       Calls BOO$MSGOUT to output an error message preceded by a
                                0253  1329 ;       new line and facility name string.
                                0253  1330 ;
                                0253  1331 ; Inputs:
                                0253  1332 ;
                                0253  1333 ;       0(SP)   - address of an ASCIZ message text
                                0253  1334 ;
                                0253  1335 ; Outputs:
                                0253  1336 ;
                                0253  1337 ;       This routine preserves all registers.
                                0253  1338 ;
                                0253  1339 ;--
                                0253  1340
                                0253  1341 BOO$FACMSG::
                      000D   30 0253  1342          BSBW    BOO$MSGOUT              ; First output facility name.
                        0A 0D 0256  1343          .ASCII  <CAR_RETURN><LINE_FEED> ; New line.
          00 38 35 54 4F  42 25 0258  1344          .ASCIZ  /%BOOT58/               ; Facility name.
                      0000   31 0260  1345          BRW     BOO$MSGOUT              ; Now output mes_ge.
                                0263  1346
                                0263  1347 ;++
                                0263  1348 ;
                                0263  1349 ; BOO$MSGOUT    - Output message on terminal
                                0263  1350 ;
                                0263  1351 ; Functional description:
                                0263  1352 ;
                                0263  1353 ;       Outputs a message on the console terminal.
                                0263  1354 ;
                                0263  1355 ; Inputs:
                                0263  1356 ;
                                0263  1357 ;       0(SP)   - the address of an ASCIZ message string
                                0263  1358 ;
                                0263  1359 ; Outputs:
                                0263  1360 ;
                                0263  1361 ;       B_ERROR_OUT     - has low bit set to indicate that a message
                                0263  1362 ;                         has been output to the terminal
                                0263  1363 ;
                                0263  1364 ;--
                                0263  1365
                                0263  1366 BOO$MSGOUT:                            ; Display message on terminal.
                          7E    7C  0263  1367          CLRQ    -(SP)               ; Create a null descriptor.
                      08  AE    DD  0265  1368          PUSHL   8(SP)               ; Pass as address of buffer.
                0000'CF   03    FB  0268  1369          CALLS   #3,BOO$READPROMPT   ; Output message.
          9E    FA00 8F   00    3A  026D  1370          LOCC    #0,#64000,@(SP)+    ; Look for zero at end of ASCIZ
                                0273  1371                                          ; message string.
                0808'CF   01    90  0273  1372          MOVB    #1,B_ERROR_OUT      ; Set a flag saying message has
                                0278  1373                                          ; been sent to terminal.
                      01  A1    17  0278  1374          JMP     1(R1)               ; Return to caller.
```

```
                        027B  1376          .SBTTL  EXECUTE_COMMAND - Execute a single command
                        027B  1377
                        027B  1378 ;++
                        027B  1379 ; Functional description:
                        027B  1380 ;
                        027B  1381 ;       Determines from the COMMAND_FLAGS settings what command is to
                        027B  1382 ;       be executed. Executes that command, clears the flag bit, and
                        027B  1383 ;       returns.
                        027B  1384 ;
                        027B  1385 ; Inputs:
                        027B  1386 ;
                        027B  1387 ;       AP points to the TPARSE parameter block, which is to be passed
                        027B  1388 ;       along intact to the command-specific action routine.
                        027B  1389 ;
                        027B  1390 ;       COMMAND_FLAGS indicates by a bit setting which command is to
                        027B  1391 ;       be executed.
                        027B  1392 ;
                        027B  1393 ; Outputs:
                        027B  1394 ;
                        027B  1395 ;       R0      - return status from the command execution routine
                        027B  1396 ;
                        027B  1397 ;       This routine BRWs to the relevant routine, and that routine
                        027B  1398 ;       returns to EXECUTE_COMMAND's caller.
                        027B  1399 ;
                        027B  1400 ;--
                        027B  1401
                   0FFC 027B  1402          .ENTRY  EXECUTE_COMMAND,-
                        027D  1403                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        027D  1404
                00   EA 027D  1405          FFS     #CMD_V_LOADCMD,-         ; Get the command bit.
                06      027F  1406                  #CMD_V_CMDFILCMD,-
           50   002C'CF 0280  1407                  COMMAND_FLAGS,R0
      00 002C'CF   50   E5 0284  1408          BBCC    R0,COMMAND_FLAGS,10$     ; Clear the bit.
                        028A  1409
                        028A  1410 10$:
                        028A  1411          CASE    R0,TYPE=B,<-             ; Case on the command.
                        028A  1412                  LOAD_FILE,-              ; Load a file.\
                        028A  1413                  START_PROGRAM,-          ; Start a program.
                        028A  1414                  EXAMINE_LOC,-            ; Examine a location.\
                        028A  1415                  DEPOSIT_DATA,-           ; Deposit data.
                        028A  1416                  PRINT_HELP,-             ; Print help text at console.
                        028A  1417                  BOOT_OP,-                ; Boot the system
                        028A  1418                  EXECUTE_FILE>            ; Execute a command file.
   06'  00   50   8F    028A        CASEB   R0,#0,S^#<<30001$-30000$>/2>-1
                        028E  30000$:
                        028E                .IRP    EP,<            LOAD_FILE,              START_PROGRAM,          EXAM
                        028E                .SIGNED_WORD    EP-30000$
                        028E                .ENDR
                00B4'   028E                .SIGNED_WORD    LOAD_FILE-30000$
                        0290
                0194'   0290                .SIGNED_WORD    START_PROGRAM-30000$
                        0292
                036E'   0292                .SIGNED_WORD    EXAMINE_LOC-30000$
                        0294
                043F'   0294                .SIGNED_WORD    DEPOSIT_DATA-30000$
                        0296
                0515'   0296                .SIGNED_WORD    PRINT_HELP-30000$
```

```
                                0298
                     0580'      0298                .SIGNED_WORD    BOOT_UP-30000$
                                029A
                     0467'      029A                .SIGNED_WORD    EXECUTE_FILE-30000$
                                029C
                                029C        30001$:
                                029C
                                029C  1419
                                029C  1420 ;
                                029C  1421 ; If none of the above, report a fatal error.
                                029C  1422 ;
                                029C  1423
                                029C  1424          MSG     <-F-Parsing error>      ; Report parsing error.
                          FFB4   30 029C            BSBW    BOO$FACMSG
65 20 67 6E 69 73 72 61 50 2D 46 2D 029F            .ASCIZ  \-F-Parsing error\
                 00 72 6F 72 72     02AB
                                02B0
                           00   02B0  1425          HALT                            ; Just halt.
```

```
                     02B1  1427          .SBTTL  SET_LOAD_FLAG - Sets the LOAD command flag.
                     02B1  1428
                     02B1  1429 ;++
                     02B1  1430 ; Functional description:
                     02B1  1431 ;
                     02B1  1432 ;       Sets the CMD_M_LOADCMD in COMMAND_FLAGS.
                     02B1  1433 ;
                     02B1  1434 ; Implicit inputs:
                     02B1  1435 ;
                     02B1  1436 ;       COMMAND_FLAGS   - holds command interpretation flags
                     02B1  1437 ;
                     02B1  1438 ; Outputs:
                     02B1  1439 ;
                     02B1  1440 ;       R0      - contains SS$_NORMAL
                     02B1  1441 ;
                     02B1  1442 ;       CMD_M_LOADCMD is set in COMMAND_FLAGS
                     02B1  1443 ;
                     02B1  1444 ;--
                     02B1  1445
               0FFC  02B1  1446          .ENTRY  SET_LOAD_FLAG,-
                     02B3  1447                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     02B3  1448
     01        C8    02B3  1449          BISL    #CMD_M_LOADCMD,-        ; Set the LOAD command flag.
002C'CF                02B5  1450                  COMMAND_FLAGS
50   01        3C    02B8  1451          MOVZWL  #SS$_NORMAL,R0          ; Set success status code.
                 04    02BB  1452          RET
```

```
                        02BC  1454          .SBTTL  SET_START_FLAG - Sets the START command flag.
                        02BC  1455
                        02BC  1456 ;++
                        02BC  1457 ; Functional description:
                        02BC  1458 ;
                        02BC  1459 ;       Sets the CMD_M_STARTCMD in COMMAND_FLAGS.
                        02BC  1460 ;
                        02BC  1461 ; Implicit inputs:
                        02BC  1462 ;
                        02BC  1463 ;       COMMAND_FLAGS   - holds command interpretation flags
                        02BC  1464 ;
                        02BC  1465 ; Outputs:
                        02BC  1466 ;
                        02BC  1467 ;       R0      - contains SS$_NORMAL
                        02BC  1468 ;
                        02BC  1469 ;       CMD_M_STARTCMD is set in COMMAND_FLAGS
                        02BC  1470 ;
                        02BC  1471 ;--
                        02BC  1472
                0FFC    02BC  1473          .ENTRY  SET_START_FLAG,-
                        02BE  1474                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        02BE  1475
          02    C8      02BE  1476          BISL    #CMD_M_STARTCMD,-       ; Set the START command flag.
002C'CF                 02C0  1477                  COMMAND_FLAGS
50    01    3C          02C3  1478          MOVZWL  #SS$_NORMAL,R0          ; Set success status code.
              04        02C6  1479          RET
```

```
                     02C7  1481          .SBTTL  SET_EXAM_FLAG - Sets the EXAMINE command flag.
                     02C7  1482
                     02C7  1483 ;++
                     02C7  1484 ; Functional description:
                     02C7  1485 ;
                     02C7  1486 ;       Sets the CMD_M_EXAMCMD in COMMAND_FLAGS.
                     02C7  1487 ;
                     02C7  1488 ; Implicit inputs:
                     02C7  1489 ;
                     02C7  1490 ;       COMMAND_FLAGS   - holds command interpretation flags
                     02C7  1491 ;
                     02C7  1492 ; Outputs:
                     02C7  1493 ;
                     02C7  1494 ;       R0      - contains SS$_NORMAL
                     02C7  1495 ;
                     02C7  1496 ;       CMD_M_EXAMCMD is set in COMMAND_FLAGS
                     02C7  1497 ;
                     02C7  1498 ;--
                     02C7  1499
                0FFC 02C7  1500          .ENTRY  SET_EXAM_FLAG,-
                     02C9  1501                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     02C9  1502
          04   C8  02C9  1503          BISL    #CMD_M_EXAMCMD,-        ; Set the EXAMINE command flag.
002C'CF             02CB  1504                  COMMAND_FLAGS
50   01   3C  02CE  1505          MOVZWL  #SS$_NORMAL,R0          ; Set success status code.
          04  02D1  1506          RET
```

```
                     02D2  1508          .SBTTL  SET_DEPOS_FLAG - Sets the DEPOSIT command flag.
                     02D2  1509
                     02D2  1510 ;++
                     02D2  1511 ; Functional description:
                     02D2  1512 ;
                     02D2  1513 ;       Sets the CMD_M_DEPOSCMD in COMMAND_FLAGS.
                     02D2  1514 ;
                     02D2  1515 ; Implicit inputs:
                     02D2  1516 ;
                     02D2  1517 ;       COMMAND_FLAGS   - holds command interpretation flags
                     02D2  1518 ;
                     02D2  1519 ; Outputs:
                     02D2  1520 ;
                     02D2  1521 ;       R0      - contains SS$_NORMAL
                     02D2  1522 ;
                     02D2  1523 ;       CMD_M_DEPOSCMD is set in COMMAND_FLAGS
                     02D2  1524 ;
                     02D2  1525 ;--
                     02D2  1526
             0FFC    02D2  1527          .ENTRY  SET_DEPOS_FLAG,-
                     02D4  1528                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     02D4  1529
       08    C8    02D4  1530          BISL    #CMD_M_DEPOSCMD,-       ; Set the DEPOSIT command flag.
002C'CF              02D6  1531                  COMMAND_FLAGS
50     01    3C    02D9  1532          MOVZWL  #SS$_NORMAL,R0          ; Set success status code.
             04    02DC  1533          RET
```

```
                        02DD  1535          .SBTTL  SET_HELP_FLAG - Sets the HELP command flag.
                        02DD  1536
                        02DD  1537 ;++
                        02DD  1538 ; Functional description:
                        02DD  1539 ;
                        02DD  1540 ;       Sets the CMD_M_HELPCMD in COMMAND_FLAGS.
                        02DD  1541 ;
                        02DD  1542 ; Implicit inputs:
                        02DD  1543 ;
                        02DD  1544 ;       COMMAND_FLAGS   - holds command interpretation flags
                        02DD  1545 ;
                        02DD  1546 ; Outputs:
                        02DD  1547 ;
                        02DD  1548 ;       R0      - contains SS$_NORMAL
                        02DD  1549 ;
                        02DD  1550 ;       CMD_M_HELPCMD is set in COMMAND_FLAGS
                        02DD  1551 ;
                        02DD  1552 ;--
                        02DD  1553
                  0FFC  02DD  1554          .ENTRY  SET_HELP_FLAG,-
                        02DF  1555                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        02DF  1556
          10    C8    02DF  1557          BISL    #CMD_M_HELPCMD,-         ; Set the HELP command flag.
002C'CF                 02E1  1558                  COMMAND_FLAGS
50    01    3C    02E4  1559          MOVZWL  #SS$_NORMAL,R0           ; Set success status code.
                04    02E7  1560          RET
```

```
                                                                J 15
                     02E8  1562          .SBTTL  SET_BOOT_FLAG - Sets the BOOT flag.
                     02E8  1563
                     02E8  1564 ;++
                     02E8  1565 ; Functional description:
                     02E8  1566 ;
                     02E8  1567 ;       Sets the CMD_M_BOOTCMD in COMMAND_FLAGS.
                     02E8  1568 ;
                     02E8  1569 ; Implicit inputs:
                     02E8  1570 ;
                     02E8  1571 ;       COMMAND_FLAGS   - holds command interpretation flags
                     02E8  1572 ;
                     02E8  1573 ; Outputs:
                     02E8  1574 ;
                     02E8  1575 ;       R0      - contains SS$_NORMAL
                     02E8  1576 ;
                     02E8  1577 ;       CMD_M_BOOTCMD is set in COMMAND_FLAGS
                     02E8  1578 ;
                     02E8  1579 ;--
                     02E8  1580
             0FFC    02E8  1581          .ENTRY  SET_BOOT_FLAG,-
                     02EA  1582                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     02EA  1583
       20    C8      02EA  1584          BISL    #CMD_M_BOOTCMD,-        ; Set the BOOT COMMAND flag.
002C'CF              02EC  1585                  COMMAND_FLAGS
50     01    3C      02EF  1586          MOVZWL  #SS$_NORMAL,R0          ; Set success status code.
             04      02F2  1587          RET
```

```
                     02F3  1589          .SBTTL  SET_CMDFIL_FLAG - Sets the COMMAND FILE flag.
                     02F3  1590
                     02F3  1591 ;++
                     02F3  1592 ; Functional description:
                     02F3  1593 ;
                     02F3  1594 ;       Sets the CMD_M_CMDFILCMD in COMMAND_FLAGS.
                     02F3  1595 ;
                     02F3  1596 ; Implicit inputs:
                     02F3  1597 ;
                     02F3  1598 ;       COMMAND_FLAGS   - holds command interpretation flags
                     02F3  1599 ;
                     02F3  1600 ; Outputs:
                     02F3  1601 ;
                     02F3  1602 ;       R0      - contains SS$_NORMAL
                     02F3  1603 ;
                     02F3  1604 ;       CMD_M_CMDFILCMD is set in COMMAND_FLAGS
                     02F3  1605 ;
                     02F3  1606 ;--
                     02F3  1607
                0FFC 02F3  1608          .ENTRY  SET_CMDFIL_FLAG,-
                     02F5  1609                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     02F5  1610
00000040 8F    C8   02F5  1611          BISL    #CMD_M_CMDFILCMD,-      ; Set the COMMAND FILE flag.
    002C'CF          02FB  1612                  COMMAND_FLAGS
    50    01    3C   02FE  1613          MOVZWL  #SS$_NORMAL,R0          ; Set success status code.
                04   0301  1614          RET
```

```
                     0302  1616          .SBTTL  MOV_FILESPEC - Pick up file specification
                     0302  1617
                     0302  1618 ;++
                     0302  1619 ; Functional description:
                     0302  1620 ;
                     0302  1621 ;       Moves the full file specification from the input string into
                     0302  1622 ;       a buffer that can be read by the file handling utilities.
                     0302  1623 ;
                     0302  1624 ; Implicit inputs:
                     0302  1625 ;
                     0302  1626 ;       AP points to the TPARSE parameter block, containing:
                     0302  1627 ;
                     0302  1628 ;       TPA$L_TOKENCNT  - number of characters in file specification
                     0302  1629 ;       TPA$L_TOKENPTR  - address of the file specification
                     0302  1630 ;
                     0302  1631 ; Implicit outputs:
                     0302  1632 ;
                     0302  1633 ;       FILE_SPEC       - holds the file specification from the input
                     0302  1634 ;                         command.
                     0302  1635 ;       FILE_DESCRIP    - the 1st word of this quadword holds the
                     0302  1636 ;                         length of the file specification string;
                     0302  1637 ;                         the 2nd longword holds the address of the
                     0302  1638 ;                         file specification string.
                     0302  1639 ;
                     0302  1640 ;--
                     0302  1641
                0FFC 0302  1642          .ENTRY  MOV_FILESPEC,-
                     0304  1643                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     0304  1644
  50   14 AC    D0  0304  1645          MOVL    TPA$L_TOKENPTR(AP),R0   ; Get address of input file
                     0308  1646                                          ; specification string.
       10 AC    28  0308  1647          MOVC3   TPA$L_TOKENCNT(AP),-    ; Move file specification
009B'CF   60        030B  1648                  (R0),FILE_SPEC          ; string into own storage.
51  004C'CF   DE    030F  1649          MOVAL   FILE_DESCRIP,R1         ; Get address of descriptor.
       10 AC    D0  0314  1650          MOVL    TPA$L_TOKENCNT(AP),-    ; Load length field.
                61  0317  1651                  DSC$W_LENGTH(R1)
     009B'CF    DE  0318  1652          MOVAL   FILE_SPEC,-             ; Load address field.
       04 A1        031C  1653                  DSC$A_POINTER(R1)
     50   01    3C  031E  1654          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                04  0321  1655          RET
```

```
                        0322  1657          .SBTTL  SAV_DEVSPEC - Pick up device specification
                        0322  1658
                        0322  1659 ;++
                        0322  1660 ; Functional description:
                        0322  1661 ;
                        0322  1662 ;       Moves a device specification from the input string into
                        0322  1663 ;       a buffer that can be read by the file handling utilities.
                        0322  1664 ;
                        0322  1665 ; Implicit inputs:
                        0322  1666 ;
                        0322  1667 ;       AP points to the TPARSE parameter block, containing:
                        0322  1668 ;
                        0322  1669 ;       TPA$L_TOKENCNT  - number of characters in file specification
                        0322  1670 ;       TPA$L_TOKENPTR  - address of the file specification
                        0322  1671 ;
                        0322  1672 ; Implicit outputs:
                        0322  1673 ;
                        0322  1674 ;       DEV_SPEC        - holds the device specification from the input
                        0322  1675 ;                         command.
                        0322  1676 ;       DEV_DESCRIP     - the 1st word of this quadword holds the
                        0322  1677 ;                         length of the device specification string;
                        0322  1678 ;                         the 2nd longword holds the address of the
                        0322  1679 ;                         device specification string.
                        0322  1680 ;
                        0322  1681 ;--
                        0322  1682
                  0FFC  0322  1683          .ENTRY  SAV_DEVSPEC,-
                        0324  1684                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0324  1685
    50   14 AC    D0  0324  1686          MOVL    TPA$L_TOKENPTR(AP),R0   ; Get address of input file
                        0328  1687                                          ; specification string.
          10 AC    28  0328  1688          MOVC3   TPA$L_TOKENCNT(AP),-    ; Move device specification
008A'CF     60        032B  1689                  (R0),DEV_SPEC           ; string into own storage.
51  008F'CF     DE  032F  1690          MOVAL   DEV_DESCRIP,R1          ; Get address of descriptor.
          10 AC    D0  0334  1691          MOVL    TPA$L_TOKENCNT(AP),-    ; Load length field.
             61        0337  1692                  DSC$W_LENGTH(R1)
       008A'CF     DE  0338  1693          MOVAL   DEV_SPEC,-              ; Load address field.
             04 A1     033C  1694                  DSC$A_POINTER(R1)
       50    01    3C  033E  1695          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                   04  0341  1696          RET
```

```
                                                N 15
                                0342  1698          .SBTTL  LOAD_FILE - Load file into memory
                                0342  1699
                                0342  1700 ;++
                                0342  1701 ; Functional description:
                                0342  1702 ;
                                0342  1703 ;       Opens a file and loads it into memory at the specified or
                                0342  1704 ;       default physical address.
                                0342  1705 ;
                                0342  1706 ; Implicit inputs:
                                0342  1707 ;
                                0342  1708 ;       START_ADDRESS   - address in which to load the file
                                0342  1709 ;       FILE_DESCRIP    - string descriptor of the file
                                0342  1710 ;       CHANNEL_NUMBER  - address into which to store a channel number
                                0342  1711 ;
                                0342  1712 ;       GOODMEM_BASE    - base address of 64K bytes of good memory
                                0342  1713 ;       FIRST_FREE_BYTE - address of first free byte past BOOT58 space;
                                0342  1714 ;                         this address is past BOOT58, BOOT58's SCB, and
                                0342  1715 ;                         3-page stack.
                                0342  1716 ;
                                0342  1717 ;--
                                0342  1718
                                0342  1719 LOAD_FILE:
                          7E    7C  0342  1720          CLRQ    -(SP)                   ; Clear space for a file
                          5E    DD  0344  1721          PUSHL   SP                      ; statistics block. Push address
                                0346  1722                                          ; on the stack for a call.
                  01A3'CF       DF  0346  1723          PUSHAL  DIREC_BUFFER            ; Send address of a buffer for
                                034A  1724                                          ; RT-11 directory segments.
                  004C'CF       DF  034A  1725          PUSHAL  FILE_DESCRIP            ; Send descriptor of file.
              0000'CF   03      FB  034E  1726          CALLS   #3,RTF$OPENFILE         ; Look up the file.
                    1D 50       E8  0353  1727          BLBS    R0,READ_FILE            ; Branch on successful lookup.
                                0356  1728          MSG     <-E-Unable to locate file>
                      FEFA      30  0356                BSBW    BOO$$ACMSG
6F 74 20 65 6C 62 61 6E 55 2D 45 2D 0359                .ASCIZ  \-E-Unable to locate file\
65 6C 69 66 20 65 74 61 63 6F 6C 20 0365
                                00  0371
                                    0372
                                04  0372  1729          RET                             ; And return.
                                    0373  1730
                                    0373  1731 ;
                                    0373  1732 ; Now read the file into memory, a block at a time. Store the number of
                                    0373  1733 ; blocks left to read in R3. The statistics block filled by RTF$OPENFILE
                                    0373  1734 ; contains the following:
                                    0373  1735 ;
                                    0373  1736 ;       +---------------------------+
                                    0373  1737 ;       |   starting LBN of file    |
                                    0373  1738 ;       +---------------------------+
                                    0373  1739 ;       | number of blocks in file  |
                                    0373  1740 ;       +---------------------------+
                                    0373  1741 ;
                                    0373  1742
                                    0373  1743 READ_FILE:                             ; Read in the file.
                    53    04 AE   D0  0373  1744          MOVL    4(SP),R3                ; Get number of blocks in file.
                54    0024'CF     D0  0377  1745          MOVL    GOODMEM_BASE,R4         ; Get base of good memory.
                54    0048'CF     D1  037C  1746          CMPL    START_ADDRESS,R4        ; Is the start address in good
                          28      1F  0381  1747          BLSSU   20$                     ; memory? Branch if not.
                      0048'CF     D1  0383  1748          CMPL    START_ADDRESS,-         ; Is the start address below
                 0000C000 E4          0387  1749                  BOOT58_OFFSET(R4)       ; the start of BOOT58 code?
```

```
                                14   1E  038C  1750          BGEQU   10$                             ; No. See if it's past BOOT58.
                          55    53   09   78  038E  1751          ASHL    #9,R3,R5                        ; Get number of bytes in file.
                          55    0048'CF   C0  0392  1752          ADDL2   START_ADDRESS,R5                ; Get final load address + 1.
                0000C000  E4    55   D1  0397  1753          CMPL    R5,BOOT58_OFFSET(R4)            ; Breaking into BOOT58?
                                3C   1B  039E  1754          BLEQU   30$                             ; No. Go load the code.
                                09   11  03A0  1755          BRB     20$                             ; Yes. Report no fit error.
                                         03A2  1756
                                         03A2  1757 10$:                                             ; See if loading past BOOT58.
                          0048'CF   D1  03A2  1758          CMPL    START_ADDRESS,-                 ; Is the start address past
                          0028'CF       03A6  1759                  FIRST_FREE_BYTE                 ; BOOT58's memory usage?
                                31   1E  03A9  1760          BGEQU   30$                             ; Yes. Branch to load code.
                                         03AB  1761
                                         03AB  1762 20$:                                             ; Invalid start address.
                                         03AB  1763          MSG     <-E-Memory in use, choose a different address>
                                FEA5     30  03AB                  BSBW    BOO$FACMSG
6E 69 20 79 72 6F 6D 65 4D 2D 45 2D      03AE                  .ASCIZ  \-E-Memory in use, choose a different address\
65 73 6F 6F 68 63 20 2C 65 73 75 20      03BA
74 6E 65 72 65 66 66 69 64 20 61 20      03C6
      00 73 73 65 72 64 64 61 20         03D2
                                         03DB
                                04   03DB  1764          RET                                     ; And return.
                                         03DC  1765
                                         03DC  1766 30$:                                             ; Load the program.
                          0048'CF   DD  03DC  1767          PUSHL   START_ADDRESS                   ; Address to receive file.
                            04 AE   DD  03E0  1768          PUSHL   4(SP)                           ; LBN to read.
                                02   DD  03E3  1769          PUSHL   #2                              ; Number of arguments.
                                         03E5  1770
                                         03E5  1771 READ_BLOCK:                                      ; Read one block.
                0000'CF   6E    FA   03E5  1772          CALLG   (SP),FIL$READ_LBN               ; Read one block.
                            18 50    E8  03EA  1773          BLBS    R0,NEXT_BLOCK                   ; Branch on sucressful read.
                                         03ED  1774          MSG     <-E-Cannot read file>
                                FE63     30  03ED                  BSBW    BOO$FACMSG
65 72 20 74 6F 6E 6E 61 43 2D 45 2D      03F0                  .ASCIZ  \-E-Cannot read file\
            00 65 6C 69 66 20 64 61      03FC
                                         0404
                                04   0404  1775          RET                                     ; Return with error status.
                                         0405  1776
                                         0405  1777 NEXT_BLOCK:                                     ; Is there another block to go?
                             01 53   F5  0405  1778          SOBGTR  R3,10$                          ; Branch forward if more blocks.
                                04   0408  1779          RET                                     ; Otherwise, return with status.
                                         0409  1780
                                         0409  1781 10$:
                             04 AE   D6  0409  1782          INCL    4(SP)                           ; Increment LBN to read by 1.
                08 AE    00000200 8F   C0  040C  1783          ADDL    #^X200,8(SP)                    ; Increase memory address by one
                                         0414  1784                                                  ; page.
                                CF   11  0414  1785          BRB     READ_BLOCK                      ; Read another block.
```

```
                        1787          .SBTTL  SET_STARTADDR - Sets a START address
                        1788
                        1789 ;++
                        1790 ; SET_STARTADDR
                        1791 ;
                        1792 ; Functional description:
                        1793 ;
                        1794 ;       Loads the start address specified in a LOAD or START command
                        1795 ;       into own storage.
                        1796 ;
                        1797 ; Inputs:
                        1798 ;
                        1799 ;       AP points to the TPARSE parameter block, containing:
                        1800 ;
                        1801 ;       TPA$L_NUMBER    - the starting address
                        1802 ;
                        1803 ; Outputs:
                        1804 ;
                        1805 ;       R1-R11 are preserved.
                        1806 ;
                        1807 ; Implicit outputs:
                        1808 ;
                        1809 ;       START_ADDRESS   - the start address value
                        1810 ;
                        1811 ;--
                        1812
                OFFC    1813          .ENTRY  SET_STARTADDR,-
                        1814                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        1815
      1C AC     D0      1816          MOVL    TPA$L_NUMBER(AP),-      ; Move the specified address
0048'CF                 1817                  START_ADDRESS           ; into own storage.
      50  01    3C      1818          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                  04    1819          RET
```

```
                     0422  1821          .SBTTL  START_PROGRAM - Start at start address
                     0422  1822
                     0422  1823 ;++
                     0422  1824 ; Functional description:
                     0422  1825 ;
                     0422  1826 ;       Restores saved or user-modified registers from the stack. Then
                     0422  1827 ;       transfers control to the address specified in this START command
                     0422  1828 ;       or in the last LOAD command.
                     0422  1829 ;
                     0422  1830 ; Implicit inputs:
                     0422  1831 ;
                     0422  1832 ;       REGISTER_TABLE  - address of saved R0-SP
                     0422  1833 ;       START_ADDRESS   - holds the start address
                     0422  1834 ;
                     0422  1835 ;--
                     0422  1836
                     0422  1837 START_PROGRAM:
5E   0040'CF   D0    0422  1838          MOVL    REGISTER_TABLE,SP       ; Call the register table the
                     0427  1839                                          ; stack top once again.
     3FFF 8F   BA    0427  1840          POPR    #^M<R0,R1,R2,R3,R4,R5,- ; Restore registers R0-FP.
                     042B  1841                  R6,R7,R8,R9,R10,R11,-
                     042B  1842                  AP,FP>
     5E   8E   D0    042B  1843          MOVL    (SP)+,SP                ; Restore saved stack top.
     0048'DF   17    042E  1844          JMP     @START_ADDRESS          ; Transfer control to start
                     0432  1845                                          ; address.
```

```
                        0432  1847          .SBTTL  SET_BYTE - Set byte display mode
                        0432  1848
                        0432  1849 ;++
                        0432  1850 ; Functional description:
                        0432  1851 ;
                        0432  1852 ;       Sets display and deposit length mode to byte.
                        0432  1853 ;
                        0432  1854 ; Implicit inputs:
                        0432  1855 ;
                        0432  1856 ;       COMMAND_FLAGS   - holds command interpretation flags
                        0432  1857 ;
                        0432  1858 ; Implicit outputs:
                        0432  1859 ;
                        0432  1860 ;       CMD_M_BYTE      - set in COMMAND_FLAGS
                        0432  1861 ;
                        0432  1862 ;--
                        0432  1863
                  0FFC  0432  1864          .ENTRY  SET_BYTE,-
                        0434  1865                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0434  1866
                        0434  1867
                    CA  0434  1868          BICL    #<CMD_M_LONG!-          ; Clear longword mode.
                        0435  1869                  CMD_M_WORD>,-           ; Clear word mode.
002C'CF  00001800 8F            0435  1870                  COMMAND_FLAGS
         00002000 8F        C8  043D  1871          BISL    #CMD_M_BYTE,-           ; Set to word mode.
             002C'CF            0443  1872                  COMMAND_FLAGS
             50   01        3C  0446  1873          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                            04  0449  1874          RET
```

```
                                044A  1876          .SBTTL  SET_WORD - Set word display mode
                                044A  1877
                                044A  1878 ;++
                                044A  1879 ; Functional description:
                                044A  1880 ;
                                044A  1881 ;       Sets display and deposit length mode to word.
                                044A  1882 ;
                                044A  1883 ; Implicit inputs:
                                044A  1884 ;
                                044A  1885 ;       COMMAND_FLAGS   - holds command interpretation flags
                                044A  1886 ;
                                044A  1887 ; Implicit outputs:
                                044A  1888 ;
                                044A  1889 ;       CMD_M_WORD      - set in COMMAND_FLAGS
                                044A  1890 ;
                                044A  1891 ;--
                                044A  1892
                           0FFC 044A  1893          .ENTRY  SET_WORD,-
                                044C  1894                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                044C  1895
                                044C  1896
                             CA 044C  1897          BICL    #<CMD_M_LONG!-          ; Clear longword mode.
                                044D  1898                  CMD_M_BYTE>,-           ; Clear byte mode.
      002C'CF    00002800 8F        044D  1899                  COMMAND_FLAGS
                 00001000 8F     C8 0455  1900          BISL    #CMD_M_WORD,-           ; Set to word mode.
                     002C'CF        045B  1901                  COMMAND_FLAGS
                     50   01     3C 045E  1902          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                                 04 0461  1903          RET
```

```
                                0462  1905          .SBTTL  SET_LONG - Set longword display mode
                                0462  1906
                                0462  1907 ;++
                                0462  1908 ; Functional description:
                                0462  1909 ;
                                0462  1910 ;       Cancels word mode for displays and deposits.
                                0462  1911 ;
                                0462  1912 ; Implicit inputs:
                                0462  1913 ;
                                0462  1914 ;       COMMAND_FLAGS   - holds command interpretation flags
                                0462  1915 ;
                                0462  1916 ; Implicit outputs:
                                0462  1917 ;
                                0462  1918 ;       CMD_M_WORD      - cleared in COMMAND_FLAGS
                                0462  1919 ;
                                0462  1920 ;--
                                0462  1921
                         0FFC   0462  1922          .ENTRY  SET_LONG,-
                                0464  1923                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                0464  1924
                           CA   0464  1925          BICL    #<CMD_M_WORD!-          ; Clear word mode.
                                0465  1926                  CMD_M_BYTE>,-           ; Clear byte mode.
      002C'CF    00003000 8F     0465  1927                  COMMAND_FLAGS
                 00000800 8F     C8  046D  1928          BISL    #CMD_M_LONG,-       ; Set long mode.
                     002C'CF         0473  1929                  COMMAND_FLAGS
                     50   01     3C  0476  1930          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                                 04  0479  1931          RET
```

```
                        047A  1933          .SBTTL  SET_PHYSICAL - Set physical address mode
                        047A  1934
                        047A  1935 ;++
                        047A  1936 ; Functional description:
                        047A  1937 ;
                        047A  1938 ;       Sets address interpretation mode to physical.
                        047A  1939 ;
                        047A  1940 ; Implicit inputs:
                        047A  1941 ;
                        047A  1942 ;       COMMAND_FLAGS   - holds command interpretation flags
                        047A  1943 ;
                        047A  1944 ; Implicit outputs:
                        047A  1945 ;
                        047A  1946 ;       CMD_M_PHYSICAL  - set in COMMAND_FLAGS
                        047A  1947 ;       CMD_M_GENERAL   - cleared in COMMAND_FLAGS
                        047A  1948 ;       CMD_M_INTERNAL  - cleared in COMMAND_FLAGS
                        047A  1949 ;
                        047A  1950 ;--
                        047A  1951
                  0FFC  047A  1952          .ENTRY  SET_PHYSICAL,-
                        047C  1953                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        047C  1954
                    CA  047C  1955          BICL    #<CMD_M_GENERAL!-       ; Clear general, internal
                        047D  1956                  CMD_M_INTERNAL>,-       ; address modes.
002C'CF    00000600 8F      047D  1957                  COMMAND_FLAGS
           00000100 8F  C8  0485  1958          BISL    #CMD_M_PHYSICAL,-       ; Set physical address mode.
               002C'CF      048B  1959                  COMMAND_FLAGS
               50   01  3C  048E  1960          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                        04  0491  1961          RET
```

```
                                0492  1963          .SBTTL  SET_GENERAL - Set general register mode
                                0492  1964
                                0492  1965 ;++
                                0492  1966 ; Functional description:
                                0492  1967 ;
                                0492  1968 ;       Sets address interpretation mode to GENERAL.
                                0492  1969 ;
                                0492  1970 ; Implicit inputs:
                                0492  1971 ;
                                0492  1972 ;       COMMAND_FLAGS   - holds command interpretation flags
                                0492  1973 ;
                                0492  1974 ; Implicit outputs:
                                0492  1975 ;
                                0492  1976 ;       CMD_M_GENERAL   - set in COMMAND_FLAGS
                                0492  1977 ;       CMD_M_PHYSICAL  - cleared in COMMAND_FLAGS
                                0492  1978 ;       CMD_M_INTERNAL  - cleared in COMMAND_FLAGS
                                0492  1979 ;
                                0492  1980 ;--
                                0492  1981
                       0FFC     0492  1982          .ENTRY  SET_GENERAL,-
                                0494  1983                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                0494  1984
                         CA     0494  1985          BICL    #<CMD_M_PHYSICAL!-      ; Clear physical, internal
                                0495  1986                  CMD_M_INTERNAL>,-       ; address modes.
          002C'CF    00000500 8F 0495  1987                  COMMAND_FLAGS
                     00000200 8F C8 049D  1988          BISL    #CMD_M_GENERAL,-        ; Set GENERAL address mode.
                         002C'CF    04A3  1989                  COMMAND_FLAGS
                         50    01 3C 04A6  1990          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                                04 04A9  1991          RET
```

```
                                04AA  1993          .SBTTL  SET_INTERNAL - Set internal processor register mode
                                04AA  1994
                                04AA  1995 ;++
                                04AA  1996 ; Functional description:
                                04AA  1997 ;
                                04AA  1998 ;       Sets address interpretation mode to INTERNAL.
                                04AA  1999 ;
                                04AA  2000 ; Implicit inputs:
                                04AA  2001 ;
                                04AA  2002 ;       COMMAND_FLAGS   - holds command interpretation flags
                                04AA  2003 ;
                                04AA  2004 ; Implicit outputs:
                                04AA  2005 ;
                                04AA  2006 ;       CMD_M_INTERNAL  - set in COMMAND_FLAGS
                                04AA  2007 ;       CMD_M_PHYSICAL  - cleared in COMMAND_FLAGS
                                04AA  2008 ;       CMD_M_GENERAL   - cleared in COMMAND_FLAGS
                                04AA  2009 ;
                                04AA  2010 ;--
                                04AA  2011
                          0FFC  04AA  2012          .ENTRY  SET_INTERNAL,-
                                04AC  2013                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                04AC  2014
                            CA  04AC  2015          BICL    #<CMD_M_PHYSICAL!-      ; Clear physical, general
                                04AD  2016                  CMD_M_GENERAL>,-        ; address modes.
002C'CF    00000300 8F          04AD  2017                  COMMAND_FLAGS
           00000400 8F      C8  04B5  2018          BISL    #CMD_M_INTERNAL,-       ; Set INTERNAL address mode.
               002C'CF          04BB  2019                  COMMAND_FLAGS
               50    01     3C  04BE  2020          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                            04  04C1  2021          RET
                                04C2  2022
```

```
                        04C2  2024          .SBTTL  EXADEP_ADDR - Pick up address argument
                        04C2  2025
                        04C2  2026 ;++
                        04C2  2027 ; Functional description:
                        04C2  2028 ;
                        04C2  2029 ;       Stores the address argument of an EXAMINE or DEPOSIT command
                        04C2  2030 ;       in own storage.
                        04C2  2031 ;
                        04C2  2032 ; Inputs:
                        04C2  2033 ;
                        04C2  2034 ;       AP points to the TPARSE parameter block, containing:
                        04C2  2035 ;
                        04C2  2036 ;       TPA$L_NUMBER    - the numeric address specified
                        04C2  2037 ;
                        04C2  2038 ; Implicit inputs:
                        04C2  2039 ;
                        04C2  2040 ;       COMMAND_FLAGS   - tells whether the value in CURRENT_ADDRESS
                        04C2  2041 ;                         is to be interpreted as:
                        04C2  2042 ;
                        04C2  2043 ;                           CMD_M_PHYSICAL - a physical address
                        04C2  2044 ;                           CMD_M_GENERAL  - a general register number
                        04C2  2045 ;                           CMD_M_INTERNAL - an internal processor
                        04C2  2046 ;                                            register number
                        04C2  2047 ;
                        04C2  2048 ; Outputs:
                        04C2  2049 ;
                        04C2  2050 ;       R1-R11 are preserved.
                        04C2  2051 ;
                        04C2  2052 ; Implicit outputs:
                        04C2  2053 ;
                        04C2  2054 ;       CURRENT_ADDRESS - holds address value
                        04C2  2055 ;
                        04C2  2056 ;--
                        04C2  2057
                    0FFC 04C2  2058          .ENTRY  EXADEP_ADDR,-
                        04C4  2059                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        04C4  2060
          50    1C AC   D0  04C4  2061          MOVL    TPA$L_NUMBER(AP),R0     ; Get numeric argument.
                08      E0  04C8  2062          BBS     #CMD_V_PHYSICAL,-       ; If physical address, no
             002C'CF        04CA  2063                  COMMAND_FLAGS,-         ; modifications necessary.
                22          04CD  2064                  SAVE_ADDRESS
                0A      E0  04CE  2065          BBS     #CMD_V_INTERNAL,-       ; If internal processor
             002C'CF        04D0  2066                  COMMAND_FLAGS,-         ; register, go interpret it.
                13          04D3  2067                  PROC_REGISTER
                        04D4  2068
                        04D4  2069 ;
                        04D4  2070 ; This is a general register number. If a valid specification, change
                        04D4  2071 ; the address to an address in the general register table.
                        04D4  2072 ;
                        04D4  2073
                50      D5  04D4  2074          TSTL    R0                      ; Register must be positive.
                21      19  04D6  2075          BLSS    BAD_REGISTER            ; Otherwise report error.
             0F 50      D1  04D8  2076          CMPL    R0,#SAVED_PC            ; Must also be between R0-PC.
                1C      14  04DB  2077          BGTR    BAD_REGISTER            ; If not, report error.
0030'CF 0040'CF 40      DE  04DD  2078          MOVAL   @REGISTER_TABLE[R0],-   ; Store address of register
                        04E5  2079                  CURRENT_ADDRESS         ; storage location.
                0E      11  04E5  2080          BRB     ADDRESS_SAVED           ; Return.
```

```
                        04E7  2081
                        04E7  2082 ;
                        04E7  2083 ; This is an internal processor register. If a valid specification,
                        04E7  2084 ; get the processor register's value and store it in own storage.
                        04E7  2085 ; Save the own storage address in CURRENT_ADDRESS.
                        04E7  2086 ;
                        04E7  2087
                        04E7  2088 PROC_REGISTER:                          ; Read a processor register.
          50    D5      04E7  2089         TSTL    R0                      ; Register must be positive.
          0E    19      04E9  2090         BLSS    BAD_REGISTER            ; Otherwise report error.
       3F 50    D1      04EB  2091         CMPL    R0,#^X3F                ; Must also be within a certain
          09    14      04EE  2092         BGTR    BAD_REGISTER            ; range. Branch if not.
                        04F0  2093
                        04F0  2094 ;
                        04F0  2095 ; Address is in R0. Save it.
                        04F0  2096 ;
                        04F0  2097
                        04F0  2098 SAVE_ADDRESS:                           ; Store physical address.
0030'CF 50    D0        04F0  2099         MOVL    R0,CURRENT_ADDRESS      ; Store address in own storage.
                        04F5  2100
                        04F5  2101 ADDRESS_SAVED:                          ; Return.
    50  01    3C        04F5  2102         MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                04      04F8  2103         RET
                        04F9  2104
                        04F9  2105 ;
                        04F9  2106 ; Bad address specification.
                        04F9  2107 ;
                        04F9  2108
                        04F9  2109 BAD_REGISTER:
                        04F9  2110         MSG     <-E-No such register>
                  FD57  30      04F9               BSBW    BOO$FACMSG
72 20 68 63 75 73 20 6F 4E 2D 45 2D   04FC               .ASCIZ  \-E-No such register\
          00 72 65 74 73 69 67 65     0508
                                      0510
                  50    D4      0510  2111         CLRL    R0                      ; Return error status
                        04      0512  2112         RET
```

```
                     0513  2114          .SBTTL  USE_LAST_ADDR - Repeat same address argument
                     0513  2115
                     0513  2116 ;++
                     0513  2117 ; Functional description:
                     0513  2118 ;
                     0513  2119 ;       Stores the last address used in a previous EXAMINE or DEPOSIT
                     0513  2120 ;       command in the address field of the current command.
                     0513  2121 ;
                     0513  2122 ; Inputs:
                     0513  2123 ;
                     0513  2124 ;       AP points to the TPARSE parameter block.
                     0513  2125 ;
                     0513  2126 ; Implicit inputs:
                     0513  2127 ;
                     0513  2128 ;       LAST_ADDRESS    - the address field of the last EXAMINE or
                     0513  2129 ;                         DEPOSIT command.
                     0513  2130 ;
                     0513  2131 ; Outputs:
                     0513  2132 ;
                     0513  2133 ;       R1-R11 are preserved.
                     0513  2134 ;
                     0513  2135 ; Implicit outputs:
                     0513  2136 ;
                     0513  2137 ;       The TPA$L_NUMBER field of the TPARSE parameter block holds the
                     0513  2138 ;       address value to be used.
                     0513  2139 ;
                     0513  2140 ;--
                     0513  2141
                 0FFC  0513  2142          .ENTRY  USE_LAST_ADDR,-
                       0515  2143                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                       0515  2144
           09    E1    0515  2145          BBC     #CMD_V_GENERAL,-        ; Branch if not general register
  0A 002C'CF           0517  2146                  COMMAND_FLAGS,10$       ;
52   0040'CF    DE    051B  2147          MOVAL   REGISTER_TABLE,R2       ; Get base addr of register table
0034'CF   52    C2    0520  2148          SUBL    R2,LAST_ADDRESS         ; Sub. base to give reg #
     0034'CF    D0    0525  2149 10$:     MOVL    LAST_ADDRESS,-          ; Move the last address used
       1C AC            0529  2150                  TPA$L_NUMBER(AP)        ; into the parameter block.
     50    01    3C    052B  2151          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                 04    052E  2152          RET
```

```
                        052F  2154          .SBTTL  USE_NEXT_ADDR - Increment address argument
                        052F  2155
                        052F  2156 ;++
                        052F  2157 ; Functional description:
                        052F  2158 ;
                        052F  2159 ;       For physical address mode, adds the current size mode (byte, word
                        052F  2160 ;       or longword, i.e., 1, 2 or 4) to the last address examined or
                        052F  2161 ;       deposited. For registers, adds 1 to the register number stored
                        052F  2162 ;       in LAST_ADDRESS.
                        052F  2163 ;
                        052F  2164 ;
                        052F  2165 ; Inputs:
                        052F  2166 ;
                        052F  2167 ;       AP points to the TPARSE parameter block.
                        052F  2168 ;
                        052F  2169 ; Implicit inputs:
                        052F  2170 ;
                        052F  2171 ;       LAST_ADDRESS    - argument of last EXAMINE or DEPOSIT command
                        052F  2172 ;       CMD_M_WORD      - if set in COMMAND_FLAGS, indicates that the
                        052F  2173 ;                         current length mode is WORD
                        052F  2174 ;
                        052F  2175 ; Outputs:
                        052F  2176 ;
                        052F  2177 ;        R1-R11 are preserved.
                        052F  2178 ;
                        052F  2179 ; Implicit outputs:
                        052F  2180 ;
                        052F  2181 ;       TPA$L_NUMBER field of the TPARSE parameter block holds the
                        052F  2182 ;       computed address.
                        052F  2183 ;
                        052F  2184 ;--
                        052F  2185
                  0FFC  052F  2186          .ENTRY  USE_NEXT_ADDR,-
                        0531  2187                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0531  2188
  51  0034'CF    D0    0531  2189          MOVL    LAST_ADDRESS,R1         ; Get last address used.
      08         E1    0536  2190          BBC     #CMD_V_PHYSICAL,-       ; If a register, go check
      002C'CF           0538  2191                  COMMAND_FLAGS,-         ; for boundary conditions.
      17                053B  2192                  CHECK_REG_MAX
      51    04    C0    053C  2193          ADDL    #4,R1                   ; Assume longword mode.
      0B    E0          053F  2194          BBS     #CMD_V_LONG,-           ; If longword mode is set,
      002C'CF           0541  2195                  COMMAND_FLAGS,-         ; just branch out to store the
      22                0544  2196                  SAVE_NEW_ADDR           ; new address in the block.
      51    02    C2    0545  2197          SUBL    #2,R1                   ; Adjust to word length.
      0C    E0          0548  2198          BBS     #CMD_V_WORD,-           ; If word mode is set,
      002C'CF           054A  2199                  COMMAND_FLAGS,-         ; just branch out to store the
      19                054D  2200                  SAVE_NEW_ADDR           ; new address in the block.
      51    01    C2    054E  2201          SUBL    #1,R1                   ; Adjust to byte mode.
      14    11          0551  2202          BRB     SAVE_NEW_ADDR           ; And go store it.
                        0553  2203
                        0553  2204 CHECK_REG_MAX:                           ; Check for maximum register #.
      51    01    C0    0553  2205          ADDL    #1,R1                   ; Increment register number.
      52    0F    D0    0556  2206          MOVL    #SAVED_PC,R2            ; Assume this is a general reg.
      09    E6          0559  2207          BBS     #CMD_V_GENERAL,-        ; If general register, branch
      002C'CF           055B  2208                  COMMAND_FLAGS,-         ; to compare new value with
      03                055E  2209                  COMPARE_REG_NUM         ; maximum register number.
      52    3F    D0    055F  2210          MOVL    #MAX_PROC_REG,R2        ; Internal register. Get maximum
```

```
                                0562  2211                                      ; value.
                                0562  2212
                                0562  2213 COMPARE_REG_NUM:                    ; Check for number too large.
                          52    0562  2214          CMPL    R1,R2             ; New number beyond maximum?
                          05 14 0565  2215          BGTR    BAD_ADDRESS       ; Yes. Report error.
                                0567  2216
                                0567  2217 SAVE_NEW_ADDR:                      ; Store the new address in the
                       1C AC 51 D0  0567  2218          MOVL    R1,TPA$L_NUMBER(AP) ; TPARSE parameter block.
                             04     056B  2219          RET                       ; Return to caller.
                                056C  2220
                                056C  2221 BAD_ADDRESS:
                                056C  2222          MSG     <-E-Invalid address specification>
                          FCE4  30  056C              BSBW    BOO$FACMSG
61 20 64 69 6C 61 76 6E 49 2D 45 2D 056F              .ASCIZ  \-E-Invalid address specification\
69 63 65 70 73 20 73 73 65 72 64 64 057B
          00 6E 6F 69 74 61 63 69 66 0587
                                0590
                          50    D4  0590  2223          CLRL    R0                ; Return error status
                                04  0592  2224          RET
```

```
                     0593  2226          .SBTTL  USE_PREVIOUS   - decrement address arguement
                     0593  2227
                     0593  2228 ;++
                     0593  2229 ; Functional description:
                     0593  2230 ;
                     0593  2231 ;       For PHYSICAL address mode, decrement the LAST_ADDRESS
                     0593  2232 ;       according to the size mode.  If in register mode, decrement
                     0593  2233 ;       the register number by 1.
                     0593  2234 ;
                     0593  2235 ; Inputs:
                     0593  2236 ;
                     0593  2237 ;       AP points to the TPARSE parameter block.
                     0593  2238 ;
                     0593  2239 ; Implicit inputs:
                     0593  2240 ;
                     0593  2241 ;       LAST_ADDRESS    - last address EXAMINEd or DISPLAYed.
                     0593  2242 ;       COMMAND_FLAS    - determines address and size modes.
                     0593  2243 ;
                     0593  2244 ; Outputs:
                     0593  2245 ;
                     0593  2246 ;       None.
                     0593  2247 ;
                     0593  2248 ; Implicit outputs:
                     0593  2249 ;
                     0593  2250 ;       TPA$_NUMBER holds the new address.
                     0593  2251 ;
                     0593  2252 ;--
                     0593  2253
                0FFC 0593  2254          .ENTRY  USE_PREVIOUS,-
                     0595  2255                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     0595  2256
     51 0034'CF  D0  0595  2257          MOVL    LAST_ADDRESS,R1         ; Get last address used
        51   01  C2  059A  2258          SUBL    #1,R1                   ; Assume register mode
             08  E1  059D  2259          BBC     #CMD_V_PHYSICAL,-       ; Branch if register mode
        002C'CF      059F  2260                  COMMAND_FLAGS,-         ;
             15      05A2  2261                  TEST_MIN_ADDR           ;
        51   03  C2  05A3  2262          SUBL    #3,R1                   ; Assume longword size
             0B  E0  05A6  2263          BBS     #CMD_V_LONG,-           ; Branch if longword size
        002C'CF      05A8  2264                  COMMAND_FLAGS,-         ;
             0C      05AB  2265                  TEST_MIN_ADDR           ;
        51   02  C0  05AC  2266          ADDL    #2,R1                   ; Adjust to word size
             0C  E0  05AF  2267          BBS     #CMD_V_WORD,-           ; Branch if word size
        002C'CF      05B1  2268                  COMMAND_FLAGS,-         ;
             03      05B4  2269                  TEST_MIN_ADDR           ;
        51   01  C0  05B5  2270          ADDL    #1,R1                   ; Adjust to byte size
                     05B8  2271
                     05B8  2272 TEST_MIN_ADDR:
                     05B8  2273 ;
                     05B8  2274 ; The new address in R1 must be greater than zero for all cases.
                     05B8  2275 ;
             51  D5  05B8  2276          TSTL    R1                      ; Is R1 < 0
             08  19  05BA  2277          BLSS    BAD_ADDR                ; Branch if yes
     1C AC   51  D0  05BC  2278          MOVL    R1,TPA$L_NUMBER(AP)     ; Save new address
        50   01  3C  05C0  2279          MOVZWL  #SS$_NORMAL,R0          ; Return success status
                 04  05C3  2280          RET
                     05C4  2281
                     05C4  2282 BAD_ADDR:
```

```
                                        05C4  2283          MSG     <-E-INVALID ADDRESS SPECIFICATION>
                                  FC8C  30  05C4                BSBW    BOO$FACMSG
41 20 44 49 4C 41 56 4E 49 2D 45 2D     05C7                .ASCIZ  \-E-INVALID ADDRESS SPECIFICATION\
49 43 45 50 53 20 53 53 45 52 44 44     05D3
         00 4E 4F 49 54 41 43 49 46     05DF
                                        05E8
                                  50    D4  05E8  2284          CLRL    R0                      ; Return error status
                                        04  05EA  2285          RET
```

```
                                05EB  2287          .SBTTL  USE_LAST_VALUE - Process "@"
                                05EB  2288
                                05EB  2289 ;++
                                05EB  2290 ; Functional description:
                                05EB  2291 ;
                                05EB  2292 ;       Retrieve the last value displayed or deposited in the last
                                05EB  2293 ;       EXAMINE or DEPOSIT command.  Sets address mode to PHYSICAL.
                                05EB  2294 ;
                                05EB  2295 ; Inputs:
                                05EB  2296 ;
                                05EB  2297 ;       AP points to the TPARSE parameter block.
                                05EB  2298 ;
                                05EB  2299 ; Outputs:
                                05EB  2300 ;
                                05EB  2301 ;       R1-R11 are preserved.
                                05EB  2302 ;
                                05EB  2303 ; Implicit outputs:
                                05EB  2304 ;
                                05EB  2305 ;       TPA$L_NUMBER field of the TPARSE parameter block holds the
                                05EB  2306 ;       last value displayed or deposited.
                                05EB  2307 ;
                                05EB  2308 ;--
                                05EB  2309
                        0FFC    05EB  2310          .ENTRY  USE_LAST_VALUE,-
                                05ED  2311                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                05ED  2312
                0038'CF   D0    05ED  2313          MOVL    LAST_VALUE,-            ; Put last value displayed in
                  1C AC         05F1  2314                  TPA$L_NUMBER(AP)        ; the parameter block.
FE82 CF   00      FB    05F3  2315          CALLS   #0,SET_PHYSICAL         ; Set address mode physical
      50  01      3C    05F8  2316          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                  04    05FB  2317          RET
```

```
                     05FC  2319          .SBTTL  EXAMINE_LOC - Display a location
                     05FC  2320
                     05FC  2321 ;++
                     05FC  2322 ; Functional description:
                     05FC  2323 ;
                     05FC  2324 ;       Displays the contents of a location on the terminal.
                     05FC  2325 ;
                     05FC  2326 ; Implicit inputs:
                     05FC  2327 ;
                     05FC  2328 ;       CURRENT_ADDRESS - the address to be examined
                     05FC  2329 ;       W_FAO_OUT_LEN   - address of a word to receive the length of
                     05FC  2330 ;                         the output string from FAO
                     05FC  2331 ;       LONG_PHY_FORMAT - address of an FAO formatting string that
                     05FC  2332 ;                         displays a hex longword
                     05FC  2333 ;       OUTPUT_FORMAT   - a writable buffer to receive an output format
                     05FC  2334 ;                         built on the fly in this routine.
                     05FC  2335 ;
                     05FC  2336 ; Outputs:
                     05FC  2337 ;
                     05FC  2338 ;       R2-R11 are preserved.
                     05FC  2339 ;
                     05FC  2340 ; Implicit outputs:
                     05FC  2341 ;
                     05FC  2342 ;       LAST_ADDRESS    - receives the address examined; if /G, value
                     05FC  2343 ;                         stored is the register #, not the address
                     05FC  2344 ;                         within the register table.
                     05FC  2345 ;       LAST_VALUE      - receives the value displayed.
                     05FC  2346 ;
                     05FC  2347 ;--
                     05FC  2348
                     05FC  2349 EXAMINE_LOC:
          07F7'CF    9F   05FC  2350          PUSHAB  OUTPUT_FORMAT           ; Push format buffer address
                     0600  2351                                              ; on the stack.
               11    28   0600  2352          MOVC3   #FORMAT_LENGTH,-        ; Move default format string
          07E6'CF         0602  2353                  LONG_PHY_FORMAT,-       ; into format buffer.
          07F7'CF         0605  2354                  OUTPUT_FORMAT
       50 07F7'CF    9E   0608  2355          MOVAB   OUTPUT_FORMAT,R0        ; Get address of format string.
               0C    E1   060D  2356          BBC     #CMD_V_WORD,-           ; If not displaying a word,
        08 002C'CF         0610  2357                  COMMAND_FLAGS,5$        ; branch.
OD A0  57583421    8F   D0   0613  2358          MOVL    #^A/!4XW/,13(R0)        ; Replace value field format.
               0D    E1   061B  2359 5$:       BBC     #CMD_V_BYTE,-           ; If not displaying a byte,
        08 002C'CF         061D  2360                  COMMAND_FLAGS,10$       ; Branch
OD A0  42583421    8F   D0   0621  2361          MOVL    #^A/!4XB/,13(R0)        ; Replace value field format.
                           0629  2362 10$:
               09    E1   0629  2363          BBC     #CMD_V_GENERAL,-        ; If not general register
        05 002C'CF         062B  2364                  COMMAND_FLAGS,20$       ; mode, branch.
     04 A0    47    8F   90   062F  2365          MOVB    #^A/G/,4(R0)            ; Replace "P" with "G".
                           0634  2366
                           0634  2367 20$:
               0A    E1   0634  2368          BBC     #CMD_V_INTERNAL,-       ; If not internal processor
        05 002C'CF         0636  2369                  COMMAND_FLAGS,30$       ; register mode, branch.
     04 A0    49    8F   90   063A  2370          MOVB    #^A/I/,4(R0)            ; Replace "P" with "I".
                           063F  2371
                           063F  2372 30$:                                    ; Put descriptors on stack.
          7E   11    9A   063F  2373          MOVZBL  #FORMAT_LENGTH,-(SP)    ; Push format length.
          07BC'CF    9F   0642  2374          PUSHAB  OUTPUT_BUFFER           ; Push buffer address on stack.
          7E   28    9A   0646  2375          MOVZBL  #OUTBUF_LENGTH,-(SP)    ; Push length too.
```

```
                 50    5E    D0  0649  2376          MOVL    SP,R0                   ; Save address of descriptors.
                                 064C  2377
                                 064C  2378 ;
                                 064C  2379 ; Push FAO arguments on stack preparatory to a CALL. In the CALL,
                                 064C  2380 ; specify an extra 4 arguments so that the return from the CALL cleans
                                 064C  2381 ; up the descriptors on the stack.
                                 064C  2382 ;
                                 064C  2383
             52  0030'CF    D0  064C  2384          MOVL    CURRENT_ADDRESS,R2      ; Get address to examine.
                    0A    E1  0651  2385          BBC     #CMD_V_INTERNAL,-       ; If not a processor register,
                002C'CF         0653  2386                  COMMAND_FLAGS,-         ; branch.
                    05          0656  2387                  EXTRACT_VALUE
                 53    52    DB  0657  2388          MFPR    R2,R3                   ; Read the processor register.
                    03    11  065A  2389          BRB     PUSH_VALUE              ; And proceed.
                                 065C  2390
                                 065C  2391 ;
                                 065C  2392 ; Just a normal physical address in R2 (either specified by user or
                                 065C  2393 ; derived for a general register).
                                 065C  2394 ;
                                 065C  2395
                                 065C  2396 EXTRACT_VALUE:                           ; Get value to display from
                 53    62    D0  065C  2397          MOVL    (R2),R3                 ; the physical address.
                                 065F  2398
                                 065F  2399 PUSH_VALUE:                              ; Put value on stack.
                    53    DD  065F  2400          PUSHL   R3                      ; Display value.
                                 0661  2401
                                 0661  2402 ;
                                 0661  2403 ; For a general register, display the register offset, not the address
                                 0661  2404 ; within the saved register table.
                                 0661  2405 ;
                                 0661  2406
                       09    E1  0661  2407          BBC     #CMD_V_GENERAL,-        ; If not a general register,
                002C'CF         0663  2408                  COMMAND_FLAGS,-         ; branch.
                       0D        0666  2409                  PUSH_ADDRESS
             54  52    0040'CF  C3  0667  2410          SUBL3   REGISTER_TABLE,R2,R4    ; Compute offset to register.
                 54    04    C6  066D  2411          DIVL    #4,R4                   ; Derive register number.
                       54    DD  0670  2412          PUSHL   R4                      ; Put register number on stack.
                       02    11  0672  2413          BRB     FINISH_STACK            ; And proceed.
                                 0674  2414
                                 0674  2415 PUSH_ADDRESS:                           ; Push a plain physical address.
                       52    DD  0674  2416          PUSHL   R2                      ; Display address of value.
                                 0676  2417
                                 0676  2418 FINISH_STACK:                           ; Complete stack setup.
                       80    7F  0676  2419          PUSHAQ  (R0)+                   ; Send output buffer descriptor.
                002C'CF         0678  2420          PUSHAW  W_FAO_OUT_LEN           ; Send location for out length.
                       60    7F  067C  2421          PUSHAQ  (R0)                    ; Send format string descriptor.
          0883'CF      09    FB  067E  2422          CALLS   #9,SYS$FAO              ; Format the output string.
                                 0683  2423
                                 0683  2424 ;
                                 0683  2425 ; Write the output string onto the console terminal by calling the
                                 0683  2426 ; read with prompt routine with a null read buffer address. This prints
                                 0683  2427 ; the prompt string and returns.
                                 0683  2428 ;
                                 0683  2429
                 50   07E4'CF    3C  0683  2430          MOVZWL  W_FAO_OUT_LEN,R0        ; Get length of output string.
                    07BC'CF40    94  0688  2431          CLRB    OUTPUT_BUFFER[R0]       ; Write a null at end of string.
                          7E     7C  068D  2432          CLRQ    -(SP)                   ; Don't ask for a read.
```

```
                                                I  1
     07BC'CF   9F   068F  2433          PUSHAB  OUTPUT_BUFFER           ; Push buffer address on stack.
0000'CF   03   FB   0693  2434          CALLS   #3,BOO$READPROMPT       ; Output string.
                    0698  2435
                    0698  2436 ;
                    0698  2437 ; Store current address in LAST_ADDRESS so that NEXT and LAST work
                    0698  2438 ; properly. For /G, only store the register number as LAST_ADDRESS.
                    0698  2439 ; Also store the value displayed. Register setup now is as follows:
                    0698  2440 ;
                    0698  2441 ;       R2      - current address
                    0698  2442 ;       R3      - value displayed
                    0698  2443 ;       R4      - general register number (if any)
                    0698  2444 ;
                    0698  2445
          09   E1   0698  2446          BBC     #CMD_V_GENERAL,-        ; If not a general register,
     002C'CF         069A  2447                  COMMAND_FLAGS,-         ; just store the address as is.
          03         069D  2448                  STORE_ADDRESS
     52   54   D0   069E  2449          MOVL    R4,R2                   ; Get register number.
                    06A1  2450
                    06A1  2451 STORE_ADDRESS:
0034'CF   52   D0   06A1  2452          MOVL    R2,LAST_ADDRESS         ; Store address away.
0038'CF   53   D0   06A6  2453          MOVL    R3,LAST_VALUE           ; Save the value displayed.
          04   06AB  2454          RET                             ; Return to caller with status
               06AC  2455                                          ; code from FAO.
```

```
                                        06AC  2457          .SBTTL  DEPOSIT_VALUE - Pick up data value
                                        06AC  2458
                                        06AC  2459 ;++
                                        06AC  2460 ; Functional description:
                                        06AC  2461 ;
                                        06AC  2462 ;       Loads the value to be deposited from a DEPOSIT command into
                                        06AC  2463 ;       own storage.
                                        06AC  2464 ;
                                        06AC  2465 ; Inputs:
                                        06AC  2466 ;
                                        06AC  2467 ;       AP points to the TPARSE parameter block, containing:
                                        06AC  2468 ;
                                        06AC  2469 ;       TPA$L_NUMBER    - the value to be deposited.
                                        06AC  2470 ;
                                        06AC  2471 ; Outputs:
                                        06AC  2472 ;
                                        06AC  2473 ;       R2-R11 are preserved.
                                        06AC  2474 ;
                                        06AC  2475 ; Implicit outputs:
                                        06AC  2476 ;
                                        06AC  2477 ;       HOLD_DEPOSIT    - holds value from DEPOSIT command.
                                        06AC  2478 ;
                                        06AC  2479 ;--
                                        06AC  2480
                                OFFC    06AC  2481          .ENTRY  DEPOSIT_VALUE,-
                                        06AE  2482                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                        06AE  2483
                      51    1C AC     D0  06AE  2484          MOVL    TPA$L_NUMBER(AP),R1     ; Get value to be deposited.
                                0C      E1  06B2  2485          BBC     #CMD_V_WORD,-           ; Branch if not word mode
                      03 002C'CF            06B4  2486                  COMMAND_FLAGS,10$       ;
                         51     51      3C  06B8  2487          MOVZWL  R1,R1                   ; Save a word.
                                0D      E1  06BB  2488 10$:     BBC     #CMD_V_BYTE,-           ; Branch if not byte mode
                      03 002C'CF            06BD  2489                  COMMAND_FLAGS,20$       ;
                         51     51      9A  06C1  2490          MOVZBL  R1,R1                   ; Save a byte.
                  0044'CF       51      D0  06C4  2491 20$:     MOVL    R1,HOLD_DEPOSIT         ; Save the value.
                         50     01      3C  06C9  2492          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                                        04  06CC  2493          RET
```

```
                        06CD  2495          .SBTTL  DEPOSIT_DATA - Store a value in memory
                        06CD  2496
                        06CD  2497 ;++
                        06CD  2498 ; Functional description:
                        06CD  2499 ;
                        06CD  2500 ;       Deposits a longword or a word of data into a specified location.
                        06CD  2501 ;
                        06CD  2502 ; Implicit inputs:
                        06CD  2503 ;
                        06CD  2504 ;       CURRENT_ADDRESS - contains the destination address
                        06CD  2505 ;       HOLD_DEPOSIT    - contains the value to be deposited
                        06CD  2506 ;
                        06CD  2507 ; Outputs:
                        06CD  2508 ;
                        06CD  2509 ;       R1-R11 are preserved.
                        06CD  2510 ;
                        06CD  2511 ; Implicit outputs:
                        06CD  2512 ;
                        06CD  2513 ;       LAST_ADDRESS    - holds destination address
                        06CD  2514 ;       LAST_VALUE      - holds value deposited
                        06CD  2515 ;
                        06CD  2516 ;--
                        06CD  2517
                        06CD  2518 DEPOSIT_DATA:
          0A   E1       06CD  2519          BBC     #CMD_V_INTERNAL,-       ; Branch if not using IPR's
09 002C'CF             06CF  2520                  COMMAND_FLAGS,10$       ;
   0044'CF   DA        06D3  2521          MTPR    HOLD_DEPOSIT,-          ; Deposit data in IPR
   0030'CF             06D7  2522                  CURRENT_ADDRESS         ;
          07   11       06DA  2523          BRB     20$                     ; Rejoin common code
   0044'CF   D0        06DC  2524 10$:     MOVL    HOLD_DEPOSIT,-          ; Deposit the value into the
   0030'DF             06E0  2525                  @CURRENT_ADDRESS        ; specified address.
   0030'CF   D0        06E3  2526 20$:     MOVL    CURRENT_ADDRESS,-       ; Save destination address.
   0034'CF             06E7  2527                  LAST_ADDRESS
   0044'CF   D0        06EA  2528          MOVL    HOLD_DEPOSIT,-          ; Save value deposited.
   0038'CF             06EE  2529                  LAST_VALUE
   50   01   3C        06F1  2530          MOVZWL  #SS$_NORMAL,R0          ; Return success status code.
                       06F4  2531          RET
```

```
                                     06F5         .SBTTL  EXECUTE_FILE - Process command file
                                     06F5
                                     06F5  ;++
                                     06F5  ; Functional description:
                                     06F5  ;
                                     06F5  ;      Opens the specified file and reads the first block into a
                                     06F5  ;      buffer. Sets a bit in the command flags longword to indicate
                                     06F5  ;      that an indirect file is being processed. The bit is used to
                                     06F5  ;      prevent indirect command file nesting.
                                     06F5  ;
                                     06F5  ; Implicit inputs:
                                     06F5  ;
                                     06F5  ;      IND_CMDBUFFER   - buffer in which to load a block of the file
                                     06F5  ;      FILE_DESCRIP    - string descriptor of the file
                                     06F5  ;
                                     06F5  ;      CMD_M_INDIRECT  - bit set in COMMAND_FLAGS if commands are
                                     06F5  ;                        already being processed from an indirect
                                     06F5  ;                        command file
                                     06F5  ;
                                     06F5  ; Outputs:
                                     06F5  ;
                                     06F5  ;      CMD_M_INDIRECT  - bit set in COMMAND_FLAGS to prevent command
                                     06F5  ;                        file nesting
                                     06F5  ;      IND_CMDBUF_PTR  - pointer to next byte in indirect command file
                                     06F5  ;                        buffer
                                     06F5  ;      IND_W_NEXT_LBN  - (1 word) next LBN to read from command file
                                     06F5  ;      IND_W_BLKS_LEFT - (1 word) number of blocks left to read in
                                     06F5  ;                        command file
                                     06F5  ;
                                     06F5  ;--
                                     06F5
                                     06F5  EXECUTE_FILE:
                           07    E1  06F5         BBC     #CMD_V_INDIRECT,-       ; If an indirect file is NOT
                   30 002C'CF        06F7                 COMMAND_FLAGS,10$       ; being processed, proceed ok.
                                     06FB         MSG     <-E-Nested command files are not permitted>
                         FB55    30  06FB         BSBW    BOO$FACMSG
6F 63 20 64 65 74 73 65 4E 2D 45 2D  06FE         .ASCIZ  \-E-Nested command files are not permitted\
20 73 65 6C 69 66 20 64 6E 61 6D 6D  070A
6D 72 65 70 20 74 6F 6E 20 65 72 61  0716
            00 64 65 74 74 69  0722
                                     0728
                         0077    31  0728         BRW     EXIT_NOW                ; Return
                                     072B
                                     072B  10$:                                   ; Accept command file.
                           7E    7C  072B         CLRQ    -(SP)                   ; Clear space for a file
                           5E    DD  072D         PUSHL   SP                      ; statistics block. Push address
                                     072F                                         ; on the stack for a call.
                      01A3'CF    DF  072F         PUSHAL  DIREC_BUFFER            ; Send address of a buffer for
                                     0733                                         ; RT-11 directory segments.
                      004C'CF    DF  0733         PUSHAL  FILE_DESCRIP            ; Send descriptor of file.
                   0000'CF   03  FB  0737         CALLS   #3,RTF$OPENFILE         ; Look up the file.
                        1E 50    E8  073C         BLBS    R0,READ_CMD_BLOCK       ; On successful read, branch.
                                     073F         MSG     <-E-Unable to locate file>
                         FB11    30  073F         BSBW    BOO$FACMSG
6F 74 20 65 6C 62 61 6E 55 2D 45 2D  0742         .ASCIZ  \-E-Unable to locate file\
65 6C 69 66 20 65 74 61 63 6F 6C 20  074E
                                 00  075A
```

```
                                                M 1
                                075B
                         45  11 075B  2580          BRB     EXIT_NOW                ; Return
                                075D  2581
                                075D  2582 ;
                                075D  2583 ; Now read the first block of the file into memory. Store
                                075D  2584 ;
                                075D  2585 ;       the number of blocks left to read,
                                075D  2586 ;       the next LBN to read,
                                075D  2587 ;       a pointer to the start of the buffer of command data
                                075D  2588 ;
                                075D  2589 ; in own storage for later reference by the routine that extracts
                                075D  2590 ; command records from the file blocks.
                                075D  2591 ;
                                075D  2592
                                075D  2593 READ_CMD_BLOCK:                          ; Read one block.
                   04 AE    01  A3 075D  2594          SUBW3   #1,4(SP),-          ; Store # blocks minus 1.
                         07A7'CF    0761  2595                  IND_W_BLKS_LEFT
               07A9'CF    6E    01  A1 0764  2596          ADDW3   #1,(SP),IND_W_NEXT_LBN ; Store second LBN in file.
                         05A3'CF    9E 076A  2597          MOVAB   IND_CMDBUFFER,-     ; Save a pointer to the
                         07A3'CF       076E  2598                  IND_CMDBUF_PTR      ; buffer.
                         05A3'CF    DF 0771  2599          PUSHAL  IND_CMDBUFFER       ; Address to receive the block.
                           04 AE    DD 0775  2600          PUSHL   4(SP)               ; Number of LBN to read.
                   0000'CF    02    FB 0778  2601          CALLS   #2,FIL$READ_LBN     ; Read 1 LBN.
                         19 50    E8 077D  2602          BLBS    R0,NORMAL_RETURN    ; Branch on successful read.
                                0780  2603          MSG     <-E-Cannot read file>
                            FADO    30 0780              BSBW    BOO$FACMSG
65 72 20 74 6F 6E 6E 61 43 2D 45 2D 0783              .ASCIZ  \-E-Cannot read file\
               00 65 6C 69 66 20 64 61 078F
                                0797
                                0797  2604
                         09  11 0797  2605          BRB     EXIT_NOW                ; Return
                                0799  2606
                                0799  2607 NORMAL_RETURN:                         ; Block is in the buffer.
                00000080 8F    C8 0799  2608          BISL    #CMD_M_INDIRECT,-   ; Set indirect command file
                         002C'CF    079F  2609                  COMMAND_FLAGS       ; bit in command flag longword.
                                07A2  2610
                                07A2  2611 EXIT_NOW:                              ; Return via RET
                            04  07A2  2612          RET
                                07A3  2613
```

```
                        07A3  2615          .SBTTL PRINT_HELP - Print an RT-11 HELP file at the console.
                        07A3  2616
                        07A3  2617 ;++
                        07A3  2618 ; Functional description:
                        07A3  2619 ;
                        07A3  2620 ;       Locate and open an RT-11 format file on the boot device, and
                        07A3  2621 ;       print that file at the console.
                        07A3  2622 ;
                        07A3  2623 ; Implicit inputs:
                        07A3  2624 ;
                        07A3  2625 ;       CMD_M_INDIRECT  - This bit is set if we are in a command file.
                        07A3  2626 ;       FILE_DESCRIP    - File descripter of file to be printed.
                        07A3  2627 ;
                        07A3  2628 ; Outputs:
                        07A3  2629 ;
                        07A3  2630 ;       None.
                        07A3  2631 ;
                        07A3  2632 ; NOTE:
                        07A3  2633 ;
                        07A3  2634 ;       HELP cannot be used from within an indirect command file.
                        07A3  2635 ;       This is because the number and size of the variables that
                        07A3  2636 ;       would have to be saved to allow this is prohibitive.
                        07A3  2637 ;
                        07A3  2638 ;--
                        07A3  2639 PRINT_HELP:                          ; Print a file
             07   E1    07A3  2640          BBC     #CMD_V_INDIRECT,-   ; Are we in a command file?
 2F 002C'CF              07A5  2641                  COMMAND_FLAGS,10$   ; Branch if not
                        07A9  2642          MSG     <-E-HELP cannot be used w/i a command file>
           FAA7    30   07A9              BSBW    BOO$FACMSG
6E 6E 61 63 20 50 4C 45 48 2D 45 2D 07AC          .ASCIZ  \-E-HELP cannot be used w/i a command file\
77 20 64 65 73 75 20 65 62 20 74 6F 07B8
64 6E 61 6D 6D 6F 63 20 61 20 69 2F 07C4
                00 65 6C 69 66 20   07D0
                                    07D6
              35    11              07D6  2643          BRB     EXIT_PRINT_HELP     ; Exit now
                                    07D8  2644 ;
                                    07D8  2645 ; Set up the help-file file descripter.
                                    07D8  2646 ;
         005E'CF    7D              07D8  2647 10$:     MOVQ    HELP_FILE_DESCRIP,- ;
         004C'CF                    07DC  2648                  FILE_DESCRIP        ;
                                    07DF  2649 ;
                                    07DF  2650 ; Call OPEN_FILE to locate and open the file.  This will also read
                                    07DF  2651 ; the first LBN of the file into memory and set CMD_V_INDIRECT.
                                    07DF  2652 ;
    087E'CF    00    FB              07DF  2653          CALLS   #0,OPEN_FILE        ; Open help file
                                    07E4  2654 ;
                                    07E4  2655 ; Now print the file one record at a time.  When the EOF is detected,
                                    07E4  2656 ; CMD_V_INDIRECT will be cleared.
                                    07E4  2657 ;
                                    07E4  2658 NEXT_RECORD:                         ; Get and print next record.
             07   E1                07E4  2659          BBC     #CMD_V_INDIRECT,-   ; Branch if EOF
         002C'CF                    07E6  2660                  COMMAND_FLAGS,-     ;
              23                    07E9  2661                  EXIT_PRINT_HELP     ;
         F9CF  CF    16             07EA  2662          JSB     GET_A_RECORD        ; Get a record from the file
           1C  50    E9             07EE  2663          BLBC    R0,EXIT_PRINT_HELP  ; Branch if error
                                    07F1  2664          ;
                                    07F1  2665          ; Print the record at the console.  A special ASCIZ string is
```

```
                        07F1  2666          ; put on the end of the buffer to cause a CRLF at the end of the
                        07F1  2667          ; record.
                        07F1  2668          ;
54   00DC'CF    9E      07F1  2669          MOVAB   COMMAND_BUFFER+1,R4     ; Get address of fisrt char
50   54    51   C1      07F6  2670          ADDL3   R1,R4,R0                ; Get address of last char
60   07B8'CF    D0      07FA  2671          MOVL    LINE_TERM,(R0)          ; Put line term past last char
           7E   7C      07FF  2672          CLRQ    -(SP)                   ; Create null descripter
           54   DD      0801  2673          PUSHL   R4                      ; Push address of first char
0000'CF    03   FB      0803  2674          CALLS   #3,BOO$READPROMPT       ; Print the record.
        02 50   E9      0808  2675          BLBC    R0,EXIT_PRINT_HELP      ; Branch if error.
           D7   11      080B  2676          BRB     NEXT_RECORD             ; Process next record.
                        080D  2677
                        080D  2678 EXIT_PRINT_HELP:
                  04    080D  2679          RET                             ; Return
```

```
                        080E  2681          .SBTTL  BOOT_UP - Boot the system
                        080E  2682 ;++
                        080E  2683 ; Functional description:
                        080E  2684 ;
                        080E  2685 ;       Boot the system using one of several generalized indirect
                        080E  2686 ;       command files.  The correct file is determined by the
                        080E  2687 ;       device specified in the BOOT command.  If no device is
                        080E  2688 ;       specified, boot using the default boot file, DEFBOO.CMD.
                        080E  2689 ;
                        080E  2690 ;       If a boot device was specified, it must be of the form
                        080E  2691 ;       DDCU:, where DD is the generic device type, C is the controller
                        080E  2692 ;       designator, and U is the unit number.  The correct boot file
                        080E  2693 ;       is DDCBOO.CMD and the unit number must be deposited in R3
                        080E  2694 ;       before executing the boot file.
                        080E  2695 ;
                        080E  2696 ;       Note that the BOOT command cannot be used w/i a command file.
                        080E  2697 ;       This is because the overhead in allowing it is prohibitive.
                        080E  2698 ;
                        080E  2699 ; Inputs:
                        080E  2700 ;
                        080E  2701 ;       COMMAND_FLAGS   - Various flags of intrest
                        080E  2702 ;       DEV_DESCRIP     - Descripter of the DDCU: string
                        080E  2703 ;       UNIT_NUMBER     - Unit number of the boot device
                        080E  2704 ;       DEFBOO_DESCRIP  - Descripter of the default boot file
                        080E  2705 ;
                        080E  2706 ; Outputs:
                        080E  2707 ;
                        080E  2708 ;       R0              - Return status
                        080E  2709 ;
                        080E  2710 ;--
                        080E  2711 BOOT_UP:                                     ; Boot the system
                  07    E1  080E  2712          BBC     #CMD_V_INDIRECT,-       ; Are we in a command file
       2F 002C'CF           0810  2713                  COMMAND_FLAGS,10$       ; Branch if yes
                           0814  2714          MSG     <-E-BOOT cannot be used w/i a command file>
                     FA3C  30    0814          BSBW    BOO$FACMSG
6E 6E 61 63 20 54 4F 4F 42 2D 45 2D 0817          .ASCIZ  \-E-BOOT cannot be used w/i a command file\
77 20 64 65 73 75 20 65 62 20 74 6F 0823
64 6E 61 6D 6D 6F 63 20 61 20 69 2F 082F
                00 65 6C 69 66 20   083B
                                    0841
                  3A    11  0841  2715          BRB     50$                     ; Exit now
                                    0843  2716 ;
                                    0843  2717 ; Decide whether to use the default command file or a special one.
                                    0843  2718 ;
                  0E    E1  0843  2719 10$:     BBC     #CMD_V_NULLDEV,-        ; Was a boot device specified?
       09 002C'CF           0845  2720                  COMMAND_FLAGS,20$       ; Branch if not
          0070'CF   7D      0849  2721          MOVQ    DEFBOO_DESCRIP,-        ; Set up boot file descripter
          004C'CF           084D  2722                  FILE_DESCRIP            ;
                  0F    11  0850  2723          BRB     30$                     ; Open the file
                            0852  2724 ;
                            0852  2725 ; Set up the boot file spec for a special boot file.
                            0852  2726 ; Only the first 3 characters of the device spec are needed.
                            0852  2727 ;
                  03    28  0852  2728 20$:     MOVC3   #3,-                    ; Length
          0093'DF           0854  2729                  @DEV_DESCRIP+4,-        ; Source
          0086'DF           0857  2730                  @BOOTFILE_DESCRIP+4     ; Destination
          0082'CF   7D      085A  2731          MOVQ    BOOTFILE_DESCRIP,-      ; Set up input for OPEN_FILE
```

```
                                                D 2
                004C'CF         085E  2732                    FILE_DESCRIP            ;
                                0861  2733 ;
                                0861  2734 ; Locate and open the boot file.
                                0861  2735 ;
087E'CF    00   FB              0861  2736 30$:    CALLS   #0,OPEN_FILE            ; Open the file
      14 50     E9              0866  2737         BLBC    R0,50$                  ; Branch if error
                                0869  2738 ;
                                0869  2739 ; If the file was opened successfully and a special boot file
                                0869  2740 ; was requested, put the unit number in R3.
                                0869  2741 ;
          0E    E4              0869  2742         BBSC    #CMD_V_NULLDEV,-        ; Was a special boot file requested?
 0B 002C'CF                     086B  2743                 COMMAND_FLAGS,40$       ; Branch if not
    50   03     9A              086F  2744         MOVZBL  #SAVED_R3,R0            ; Get index of saved R3 w/i table
   0097'CF     D0              0872  2745         MOVL    UNIT_NUMBER,-           ; Put unit number in saved R3
  0040'DF40                     0876  2746                 @REGISTER_TABLE[R0]     ;
    50   01     3C              087A  2747 40$:    MOVZWL  #SS$_NORMAL,R0          ; Put return status in R0
                04              087D  2748 50$:    RET                             ; Return
```

```
                    087E  2750
                    087E  2751          .SBTTL  OPEN_FILE - Locate and open an Rt-11 file.
                    087E  2752 ;++
                    087E  2753 ;
                    087E  2754 ; This routine codes around a messy procedure interface.
                    087E  2755 ; This is only a dummy routine that branches to EXECUTE_FILE
                    087E  2756 ; and allows it to return to OPEN_FILE's caller via a RET.
                    087E  2757 ;
                    087E  2758 ;--
                    087E  2759
               0000 087E  2760          .ENTRY  OPEN_FILE,^M<>
FE72             31 0880  2761          BRW     EXECUTE_FILE
```

```
                        0883  2763          .SBTTL  Stand alone I/O support
                        0883  2764
                        0883  2765 ;++
                        0883  2766 ;
                        0883  2767 ; SYS$FAO       - formats an output string
                        0883  2768 ;
                        0883  2769 ; Functional description:
                        0883  2770 ;
                        0883  2771 ;       Calls the system service directly instead of through a vector.
                        0883  2772 ;
                        0883  2773 ;--
                        0883  2774
                   0FFC 0883  2775          .ENTRY  SYS$FAO,-
                        0885  2776                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0885  2777
                F77A' 31 0885  2778          BRW     EXE$FAO+2                       ; Branch past entry mask.
```

```
                                0888  2780          .SBTTL  XDELTA Definitions and a fault handler
                                0888  2781
                                0888  2782 ;
                                0888  2783 ; Define handlers needed by XDELTA.
                                0888  2784 ;
                                0888  2785
                                0888  2786          .ALIGN  LONG                    ; All handlers longword-aligned.
                                0888  2787
                    00000001    0888  2788          .IF     NE,SADEBUG              ; If XDELTA is included...
                                0888  2789
                                0888  2790 EXE$ACVIOLAT::                          ; Access violation vector.
                                0888  2791 EXE$BREAK::                            ; BPB vector.
                                0888  2792 EXE$ROPRAND::                          ; Reserved operand vector.
                                0888  2793 EXE$TBIT::                             ; Trace trap vector.
                                0888  2794 EXE$GB_CPUTYPE::                       ; Defines CPU to be an 11/750
                    00000002    0888  2795          .LONG   PR$_SID_TYP750          ;
                                088C  2796 MMG$PAGEFAULT::                        ; Pagefault exception vector.
                                088C  2797
                                088C  2798          .ENDC                           ; End of XDELTA conditional.
                                088C  2799
                                088C  2800 ;
                                088C  2801 ; Fault handler for unexpected exception conditions during BOOT58.
                                088C  2802 ;
                                088C  2803
                                088C  2804 BOOT_FAULT:                            ; Handler for most of SCB.
                                088C  2805
                    00000001    088C  2806          .IF     NE,SADEBUG              ; If XDELTA included...
                  F7C0 CF   16  088C  2807          JSB     INI$BRK                 ; Call XDELTA
                                0890  2808          .ENDC                           ; End conditional
                                0890  2809
                                0890  2810          MSG     </-F-Unexpected Exception/>
                      F9C0   30 0890              BSBW    BOO$FACMSG
74 63 65 70 78 65 6E 55 2D 46 2D 2F 0893              .ASCIZ  \/-F-Unexpected Exception/\
6E 6F 69 74 70 65 63 78 45 20 64 65 089F
                         00 2F 08AB
                               08AD
                08 0809'CF   E9 08AD  2811          BLBC    B_ENVIRON_SAFE,10$      ; Branch forward if this is
                               08B2  2812                                          ; a fatal exception.
             5E   080A'CF    D0 08B2  2813          MOVL    L_OLD_STACKP,SP         ; Restore SP.
                    F7DF    31 08B7  2814          BRW     READ_COMMAND            ; Go try again.
                               08BA  2815
                               08BA  2816 10$:                                    ; Fatal exception.
                            00 08BA  2817          HALT                            ; Halt processor.
                               08BB  2818
                               08BB  2819 ;
                               08BB  2820 ; Labels required for XDELTA.
                               08BB  2821 ;
                               08BB  2822
                    00000001   08BB  2823          .IF     NE,SADEBUG              ; If XDELTA is included...
                               08BB  2824
                               08BB  2825 SYSL$CLRSBIA::
                               08BB  2826 INI$RDONLY::                            ; Dummy change protection
                               08BB  2827 INI$WRITABLE::                          ; routines.
                            05 08BB  2828          RSB                             ; Just return.
                               08BC  2829
                               08BC  2830 EXE$GL_FLAGS::                          ; Dummy flags longword.
                               08BC  2831 EXE$GL_SCB::                            ; Dummy SCB address pointer.
```

```
00000000  08BC  2832 EXE$V_SIMULATOR == 0                    ; This is not a simulator.
          08BC  2833
          08BC  2834 PFN$AB_STATE::
          08BC  2835 PFN$AB_TYPE::
          08BC  2836 PFN$AL_BAK::
          08BC  2837 PFN$AL_PTE::
          08BC  2838 PFN$AW_BLINK::
          08BC  2839 PFN$AW_FLINK::
          08BC  2840 PFN$AX_BLINK::
          08BC  2841 PFN$AX_FLINK::
          08BC  2842 PFN$AW_REFCNT::
          08BC  2843 PFN$AW_SWPVBN::
          08BC  2844
          08BC  2845 SYS$IOBASE::
          08BC  2846 SCH$GL_CURPCB::
          08BC  2847 SCH$GL_PCBVEC::
          08BC  2848
          08BC  2849 XDS$GT_WORD_PFN::
          08BC  2850 XDS$INIT::
00000000  08BC  2851         .LONG   0
          08C0  2852
          08C0  2853         .ENDC                          ; End of XDELTA conditional.
          08C0  2854
          08C0  2855 ;
          08C0  2856 ; Dummy routine for SYS$ASCTOID. This routine is called by TPARSE for
          08C0  2857 ; the TPA$_IDENT item, which is not used by BOOT58.
          08C0  2858 ;
          08C0  2859
          08C0  2860 SYS$UNWIND::
          08C0  2861 SYS$ASCTIM::
          08C0  2862 SYS$IDTOASC::
          08C0  2863 SYS$ASCTOID::
          08C0  2864 SYS$FILESCAN::
     0000 08C0  2865         .WORD   0                      ; Null entry mask
   50  D4 08C2  2866         CLRL    R0                     ; Return LBC to indicate failure
       04 08C4  2867         RET
          08C5  2868
          08C5  2869 ;
          08C5  2870 ; Labels required for RMSCONIO.
          08C5  2871 ;
          08C5  2872
00000000  08C5  2873         .IF     NE,TSDEBUG             ; If DEBUG is being linked...
          08C5  2874
          08C5  2875 BOO$AB_PRMBUF::
          08C5  2876 BOO$A_PRMBLK::
          08C5  2877 BOO$GQ_CMDESC::                        ; String descriptor for READ.
          08C5  2878         .LONG   0,RMS_BUFFER           ; Input buffer for RMS read.
          08C5  2879
          08C5  2880 RMS_BUFFER:                            ; Input buffer for RMS.
          08C5  2881         .BYTE   100
          08C5  2882
          08C5  2883 RMS_BUF_LENGTH  = .-RMS_BUFFER         ; Length of RMS buffer.
          08C5  2884
          08C5  2885         .ENDC
```

```
                        08C5  2887          .SBTTL  Declarations that must appear at the end
                        08C5  2888
                        08C5  2889 ;
                        08C5  2890 ; If XDELTA is linked, align end of BOOT58 on a page boundary.
                        08C5  2891 ;
                        08C5  2892
00000001                08C5  2893          .IF     NE,SADEBUG
                        08C5  2894          .ALIGN  LONG
                        08C8  2895          .ENDC
                        08C8  2896
                        08C8  2897          .NOSHOW EXPANSIONS
                        08C8  2898
                        08C8  2899          .END    BOOT58                  ; Transfer address.
```

```
$$$CNT                          = 00000003
$$$FLG                          = FFFFFFFF
$$$KEY                          = 00000006
$$$KFG                          = FFFFFFFF
$$$MOD                          = 00000002
$$$TMP                          = 0000002C R      06
$$KEYTAB                        = 00000000 R      05
ADDRESS_SAVED                     000004F5 R      07
ANY_VALUE                         000000E6 R      04
BAD_ADDR                          000005C4 R      07
BAD_ADDRESS                       0000056C R      07
BAD_REGISTER                      000004F9 R      07
BIT...                          = 00000010
BOO$FACMSG                        00000253 RG     07
BOO$MSGOUT                        00000263 R      07
BOO$READPROMPT                    ******** **   X 07
BOOT58                            00000057 RG     07
BOOT58_OFFSET                   = 0000C000
BOOT58_TABLE                      00000000 RG     04
BOOTCMD                           00000138 R      04
BOOTFILE_DESCRIP                  00000082 R      02
BOOTFILE_SPEC                     00000078 R      02
BOOTHIGH                          00000000 R      03
BOOT_FAULT                        0000088C R      07
BOOT_UP                           0000080E R      07
BYPASS_COMMENT                    0000012C R      07
B_ENVIRON_SAFE                    00000809 R      02
B_ERROR_OUT                       00000808 R      02
CAR_RETURN                      = 0000000D
CHECK_REG_MAX                     00000553 R      07
CLEAR_ERRBIT                      000001AD R      07
CMDFILE                           00000174 R      04
CMD_BUFFER_SIZE                 = 000000C4
CMD_M_BOOTCMD                   = 00000020
CMD_M_BYTE                      = 00002000
CMD_M_CMDFILCMD                 = 00000040
CMD_M_DEPOSCMD                  = 00000008
CMD_M_EXAMCMD                   = 00000004
CMD_M_GENERAL                   = 00000200
CMD_M_HELPCMD                   = 00000010
CMD_M_INDIRECT                  = 00000080
CMD_M_INTERNAL                  = 00000400
CMD_M_LOADCMD                   = 00000001
CMD_M_LONG                      = 00000800
CMD_M_NULLDEV                   = 00004000
CMD_M_PHYSICAL                  = 00000100
CMD_M_STARTCMD                  = 00000002
CMD_M_UNATTEND                  = 00008000
CMD_M_WORD                      = 00001000
CMD_STRING_SIZE                 = 00000400
CMD_V_BOOTCMD                   = 00000005
CMD_V_BYTE                      = 0000000D
CMD_V_CMDFILCMD                 = 00000006
CMD_V_DEPOSCMD                  = 00000003
CMD_V_EXAMCMD                   = 00000002
CMD_V_GENERAL                   = 00000009
CMD_V_HELPCMD                   = 00000004
CMD_V_INDIRECT                  = 00000007
CMD_V_INTERNAL                  = 0000000A
CMD_V_LOADCMD                   = 00000000
CMD_V_LONG                      = 0000000B
CMD_V_NULLDEV                   = 0000000E
CMD_V_PHYSICAL                  = 00000008
CMD_V_STARTCMD                  = 00000001
CMD_V_UNATTEND                  = 0000000F
CMD_V_WORD                      = 0000000C
COMMAND_BUFFER                    000000DB R      02
COMMAND_FLAGS                     0000002C R      02
COMMAND_STRING                    000001A3 R      02
COMPARE_REG_NUM                   00000562 R      07
CONTROLLER                        0000016C R      04
CURRENT_ADDRESS                   00000030 R      02
DEFBOO_DESCRIP                    00000070 R      02
DEFBOO_SPEC                       00000066 R      02
DEPOSITCMD                        0000007A R      04
DEPOSIT_DATA                      000006CD R      07
DEPOSIT_VALUE                     000006AC RG     07
DEVICE_TYPE                       00000166 R      04
DEVSPEC                           00000150 R      04
DEV_DESCRIP                       0000008F R      02
DEV_SPEC                          0000008A R      02
DIREC_BUFFER                      000001A3 R      02
DRIVER_SUBROUT                    0000080E RG     02
DSC$A_POINTER                   = 00000004
DSC$W_LENGTH                    = 00000000
END_OF_FILE                       0000023A R      07
END_OF_LINE                       00000165 R      07
END_OF_RECORD                     00000240 R      07
EXADEP_ADDR                       000004C2 RG     07
EXADEP_QUAL                       00000092 R      04
EXAMINECMD                        0000006A R      04
EXAMINE_LOC                       000005FC R      07
EXE$ACVIOLAT                      00000888 RG     07
EXE$BREAK                         00000888 RG     07
EXE$FAO                           ******** **   X 07
EXE$GB_CPUTYPE                    00000888 RG     07
EXE$GL_FLAGS                      000008BC RG     07
EXE$GL_SCB                        000008BC RG     07
EXE$MCHKVEC                     = 00000004 RG     03
EXE$ROPRAND                       00000888 RG     07
EXE$TBIT                          00000888 RG     07
EXE$V_SIMULATOR                 = 00000000  G
EXECUTE_COMMAND                   0000027B RG     07
EXECUTE_FILE                      000006F5 R      07
EXIT_GETRECORD                    00000252 R      07
EXIT_NOW                          000007A2 R      07
EXIT_PRINT_HELP                   0000080D R      07
EXPLICIT_VALUE                    000000F4 R      04
EXTRACT_VALUE                     0000065C R      07
FIL$READ_LBN                      ******** **   X 07
FILESPEC                          00000112 R      04
FILE_DESCRIP                      0000004C R      02
FILE_SPEC                         0000009B R      02
FILL_SCB                          00000022 R      07
```

```
FINISH_STACK                    00000676 R      07
FIRST_FREE_BYTE                 00000028 R      02
FORMAT_LENGTH                 = 00000011
FORM_FEED                     = 0000000C
GET_A_CHARACTER                 00000119 R      07
GET_A_RECORD                    000001BD R      07
GET_CHAR                        000001CE R      07
GOODMEM_BASE                    00000024 R      02
GOT_A_LINE                      0000010B R      07
HELPCMD                         00000132 R      04
HELP_FILE_DESCRIP               0000005E R      02
HELP_FILE_SPEC                  00000054 R      02
HOLD_DEPOSIT                    00000044 R      02
HYPHEN_IN_LINE                  00000139 R      07
IND_CMDBUFFER                   000005A3 R      02
IND_CMDBUF_PTR                  000007A3 R      02
IND_W_BLKS_LEFT                 000007A7 R      02
IND_W_NEXT_LBN                  000007A9 R      02
INI$BRK                         00000050 RG     07
INI$RDONLY                      000008BB RG     07
INI$WRITABLE                    000008BB RG     07
IN_COMMAND_FILE                 000007AD R      02
KEY_TABLE                       00000000 RG     05
LAST_ADDRESS                    00000034 R      02
LAST_VALUE                      00000038 R      02
LIB$TPARSE                      ********   X    07
LINE_FEED                     = 0000000A
LINE_TERM                       000007B8 R      02
LOADCMD                         0000004A R      04
LOAD_FILE                       00000342 R      07
LOAD_QUAL                       00000122 R      04
LOC_QUAL                        0000009E R      04
LONG_PHY_FORMAT                 000007E6 R      02
LOOK_FOR_BLANK                  00000148 R      07
L_OLD_STACKP                    0000080A R      02
MAX_PROC_REG                  = 0000003F
MMG$PAGEFAULT                   0000088C RG     07
MORE_CHARS                      00000127 R      07
MOV_FILESPEC                    00000302 RG     07
NEXT_BLOCK                      00000405 P      07
NEXT_CHAR                       00000162 R      07
NEXT_COMMAND                    000001B1 R      07
NEXT_RECORD                     000007E4 R      07
NOBRK                           00000051 R      07
NORMAL_RETURN                   00000799 R      07
NULL                          = 00000000
NUMBER                          000000DA R      04
NUM_OF_CMDS                   = 00000007
OPEN_FILE                       0000087E RG     07
OUTBUF_LENGTH                 = 00000028
OUTPUT_BUFFER                   000007BC R      02
OUTPUT_FORMAT                   000007F7 R      02
PARAM_BLOCK                     00000000 R      02
PARSE_COMMAND                   0000017C R      07
PFN$AB_STATE                    000008BC RG     07
PFN$AB_TYPE                     000008BC RG     07
PFN$AL_BAK                      000008BC RG     07
PFN$AL_PTE                      000008BC RG     07
PFN$AW_BLINK                    000008BC RG     07
PFN$AW_FLINK                    000008BC RG     07
PFN$AW_REFCNT                   000008BC RG     07
PFN$AW_SWPVBN                   000008BC RG     07
PFN$AX_BLINK                    000008BC RG     07
PFN$AX_FLINK                    000008BC RG     07
PR$_SCBB                      = 00000011
PR$_SID_TYP750                = 00000002
PRINT_HELP                      000007A3 R      07
PROC_REGISTER                   000004E7 R      07
PROMPT_STRING                   000007AE R      02
PUSH_ADDRESS                    00000674 R      07
PUSH_VALUE                      0000065F R      07
READ_A_LINE                     000000A7 R      07
READ_BLOCK                      000003E5 R      07
READ_CMD_BLOCK                  0000075D R      07
READ_COMMAND                    00000099 R      07
READ_FILE                       00000373 R      07
REGISTER_TABLE                  00000040 R      02
RETURN_RECORD                   0000024F R      07
RPB$M_MPM                     = 00000800
RPB$M_NODEFBOO                = 00000800
RPB$V_BOOBPT                  = 00000005
RPB$V_MPM                     = 0000000B
RPB$V_NODEFBOO                = 0000000B
RTF$OPENFILE                    ********   X    07
SADEBUG                       = 00000001
SAENVIRON                     = 00000001
SAVED_AP                      = 0000000C
SAVED_FP                      = 0000000D
SAVED_PC                      = 0000000F
SAVED_R0                      = 00000000
SAVED_R1                      = 00000001
SAVED_R10                     = 0000000A
SAVED_R11                     = 0000000B
SAVED_R2                      = 00000002
SAVED_R3                      = 00000003
SAVED_R4                      = 00000004
SAVED_R5                      = 00000005
SAVED_R6                      = 00000006
SAVED_R7                      = 00000007
SAVED_R8                      = 00000008
SAVED_R9                      = 00000009
SAVED_SP                      = 0000000E
SAVED_STATUS                    000007AB R      02
SAVED_VALUE                     0000003C R      02
SAVE_ADDRESS                    000004F0 R      07
SAVE_NEW_ADDR                   00000567 R      07
SAV_DEVSPEC                     00000322 RG     07
SCH$GL_CURPCB                   000008BC RG     07
SCH$GL_PCBVEC                   000008BC RG     07
SETUP_BOOT58                    00000000 R      07
SET_BOOT_FLAG                   000002E8 RG     07
SET_BYTE                        00000432 RG     07
SET_CMDFIL_FLAG                 000002F3 RG     07
SET_DEPOS_FLAG                  000002D2 RG     07
```

```
SET_EXAM_FLAG                   000002C7 RG    07
SET_GENERAL                     00000492 RG    07
SET_HELP_FLAG                   000002DD RG    07
SET_INTERNAL                    000004AA RG    07
SET_LOAD_FLAG                   000002B1 RG    07
SET_LONG                        00000462 RG    07
SET_PHYSICAL                    0000047A RG    07
SET_STARTADDR                   00000416 RG    07
SET_START_FLAG                  000002BC RG    07
SET_WORD                        0000044A RG    07
SIZ...                        = 00000001
SIZE_QUAL                       000000BC R     04
SKIP_COMMENT                    0000016F R     07
SS$_NORMAL                    = 00000001
STARTCMD                        0000005E R     04
START_ADDRESS                   00000048 R     02
START_PROGRAM                   00000422 R     07
STORE_ADDRESS                   000006A1 R     07
SYS$ASCTIM                      000008C0 RG    07
SYS$ASCTOID                     000008C0 RG    07
SYS$FAO                         00000883 RG    07
SYS$FILESCAN                    000008C0 RG    07
SYS$IDTOASC                     000008C0 RG    07
SYS$IOBASE                      000008BC RG    07
SYS$UNWIND                      000008C0 RG    07
SYSL$CLRSBIA                    000008BB RG    07
TEST_MIN_ADDR                   000005B8 R     07
TPA$K_COUNTO                  = 00000008
TPA$K_LENGTHO                 = 00000024
TPA$L_COUNT                   = 00000000
TPA$L_NUMBER                  = 0000001C
TPA$L_OPTIONS                 = 00000004
TPA$L_STRINGCNT               = 00000008
TPA$L_STRINGPTR               = 0000000C
TPA$L_TOKENCNT                = 00000010
TPA$L_TOKENPTR                = 00000014
TPA$M_ABBREV                  = 00000002
TPA$_ALPHA                    = 000001EE
TPA$_ANY                      = 000001ED
TPA$_BLANK                    = 000001F2
TPA$_DECIMAL                  = 000001F3
TPA$_DIGIT                    = 000001EF
TPA$_EOS                      = 000001F7
TPA$_EXIT                     = FFFFFFFF
TPA$_FAIL                     = FFFFFFFE
TPA$_FILESPEC                 = 000001EA
TPA$_HEX                      = 000001F5
TPA$_IDENT                    = 000001EC
TPA$_KEYWORD                  = 00000100
TPA$_LAMBDA                   = 000001F6
TPA$_MAXKEY                   = 000000DC
TPA$_OCTAL                    = 000001F4
TPA$_STRING                   = 000001F0
TPA$_SUBXPR                   = 000001F8
TPA$_SYMBOL                   = 000001F1
TPA$_UIC                      = 000001EB
TSDEBUG                       = 00000000
UNIT_NUMBER                     00000097 R     02
USE_LAST_ADDR                   00000513 RG    07
USE_LAST_VALUE                  000005EB RG    07
USE_NEXT_ADDR                   0000052F RG    07
USE_PREVIOUS                    00000593 RG    07
VALID_COMMAND                   0000000E R     04
VERTICAL_TAB                  = 0000000B
W_FAO_OUT_LEN                   000007E4 R     02
XDELBPT                         ******** X     07
XDELIBRK                        ******** X     07
XDELTBIT                        ******** X     07
XDS$GT_WORD_PFN                 000008BC RG    07
XDS$INIT                        000008BC RG    07
```

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
BOOT58_DATA	00000812 (2066.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
Z99BOOT	00000000 (0.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
_LIB$STATE$	00000180 (384.)	04 (4.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE
_LIB$KEY0$	0000000E (14.)	05 (5.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC WORD
_LIB$KEY1$	00000032 (50.)	06 (6.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC WORD
$$$$00BOOT	000008C8 (2248.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:00.34
Command processing	109	00:00:00.61	00:00:01.81
Pass 1	565	00:00:28.73	00:01:00.16
Symbol table sort	0	00:00:01.55	00:00:02.62
Pass 2	412	00:00:09.16	00:00:17.06
Symbol table output	0	00:00:00.23	00:00:00.39
Psect synopsis output	0	00:00:00.04	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1117	00:00:40.42	00:01:22.43

The working set limit was 2000 pages.
164925 bytes (323 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1015 non-local and 29 local symbols.
2899 source lines were read in Pass 1, producing 111 object records in Pass 2.
34 pages of virtual memory were used to define 28 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
_255DUA28:[SYS.OBJ]LIB.MLB;1	2
_255DUA28:[SYSLIB]STARLET.MLB;2	13
TOTALS (all libraries)	15

1106 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:BOOT58/OBJ=OBJ$:BOOT58 MSRC$:BOOT58/UPDATE=(ENH$:BOOT58)+EXECML$/LIB+LIB$:BOOTS.MLB/LIB
```

BOOTS
MAR

AUTOCONFG
LIS

WRITEBOOT
MAP

BOOCVTATB
LIS

VMB
MAP

SYSGEN
MAP

ACTIMAGE
LIS

BOOT58
LIS

BOOTDEF
SDL

CONFIG
LIS

BTMEM855
LIS

BTMEM790
LIS

CONFIGURE
LIS

BOOTDEF
LIS

BOOTIO
LIS

CONFIGMN
LIS

BOOTDRIVR
LIS

BTMEM730
LIS

BTMEM750
LIS

BTMEM780
LIS

BOOTBLOCK
LIS