```
   SSSSSSSSSSSS   DDDDDDDDDDDD        AAAAAAAAA
   SSSSSSSSSSSS   DDDDDDDDDDDD        AAAAAAAAA
   SSSSSSSSSSSS   DDDDDDDDDDDD        AAAAAAAAA
SSS               DDD         DDD  AAA         AAA
SSS               DDD         DDD  AAA         AAA
SSS               DDD         DDD  AAA         AAA
SSS               DDD         DDD  AAA         AAA
SSS               DDD         DDD  AAA         AAA
SSS               DDD         DDD  AAA         AAA
   SSSSSSSSS      DDD         DDD  AAA         AAA
   SSSSSSSSS      DDD         DDD  AAA         AAA
   SSSSSSSSS      DDD         DDD  AAA         AAA
            SSS   DDD         DDD  AAAAAAAAAAAAAAA
            SSS   DDD         DDD  AAAAAAAAAAAAAAA
            SSS   DDD         DDD  AAAAAAAAAAAAAAA
            SSS   DDD         DDD  AAA         AAA
            SSS   DDD         DDD  AAA         AAA
            SSS   DDD         DDD  AAA         AAA
SSSSSSSSSSSS      DDDDDDDDDDDD     AAA         AAA
SSSSSSSSSSSS      DDDDDDDDDDDD     AAA         AAA
SSSSSSSSSSSS      DDDDDDDDDDDD     AAA         AAA
```

FILEID**SYMBOLS

B 4

```
  SSSSSSSS  YY      YY  MM      MM  BBBBBBBB      OOOOOO    LL            SSSSSSSS
  SSSSSSSS  YY      YY  MM      MM  BBBBBBBB      OOOOOO    LL            SSSSSSSS
SS          YY      YY  MMMM  MMMM  BB      BB  OO      OO  LL          SS
SS          YY      YY  MMMM  MMMM  BB      BB  OO      OO  LL          SS
SS            YY  YY    MM  MM  MM  BB      BB  OO      OO  LL          SS
SS            YY  YY    MM  MM  MM  BB      BB  OO      OO  LL          SS
  SSSSSS        YY      MM      MM  BBBBBBBB    OO      OO  LL            SSSSSS
  SSSSSS        YY      MM      MM  BBBBBBBB    OO      OO  LL            SSSSSS
        SS      YY      MM      MM  BB      BB  OO      OO  LL                  SS
        SS      YY      MM      MM  BB      BB  OO      OO  LL                  SS
        SS      YY      MM      MM  BB      BB  OO      OO  LL                  SS      ....
        SS      YY      MM      MM  BB      BB  OO      OO  LL                  SS      ....
SSSSSSSS        YY      MM      MM  BBBBBBBB      OOOOOO    LLLLLLLLLL  SSSSSSSS        ....
SSSSSSSS        YY      MM      MM  BBBBBBBB      OOOOOO    LLLLLLLLLL  SSSSSSSS        ....

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        .TITLE  SYMBOLS SYSTEM SYMBOL TABLE ROUTINES
0000        .SBTTL  COPYRIGHT NOTICE
0000        .IDENT  'V04-000'
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
```

```
0000          .SBTTL  PROGRAM DESCRIPTION
0000    ;++
0000    ; FACILITY
0000    ;
0000    ;       SYSTEM DUMP ANALYZER
0000    ;
0000    ; ABSTRACT
0000    ;
0000    ;       ROUTINE TO READ THE SYSTEM SYMBOL TABLE AND
0000    ;       MANIPULATE THE INTERNAL SORTED SYMBOL TABLE.
0000    ;
0000    ; ENVIRONMENT
0000    ;
0000    ;       NATIVE MODE, USER MODE
0000    ;
0000    ; AUTHOR
0000    ;
0000    ;       TIM HALVORSEN, JULY 1978
0000    ;
0000    ; MODIFIED BY
0000    ;
0000    ;       V03-010 EMD0100         Ellen M. Dusseault      09-May-1984
0000    ;               Remove temporary fix for formatting the UCB which
0000    ;               was to increase the max number of symbols to format.
0000    ;               This round do it the right way.  Determine the device
0000    ;               and only display its ucb extensions instead of all
0000    ;               device extensions.
0000    ;               Add CLU$GL_CLUB to required symbol table.
0000    ;               Add SCS symbols to required symbol table to support
0000    ;               new SCS displays.
0000    ;
0000    ;       V03-009 EMD0085         Ellen M. Dusseault      15-Apr-1984
0000    ;               Modify routine, SORT_OFFSETS, to ignore all constant
0000    ;               symbols except for the length symbol.  Fix problem
0000    ;               with formatting the UCB (increase max number of symbols
0000    ;               to format).
0000    ;
0000    ;       V03-008 EMD0078         Ellen M. Dusseault      10-Apr-1984
0000    ;               Define symbols to represent the base addresses of
0000    ;               code in nonpaged pool (such as sysloa,scsloa).
0000    ;
0000    ;       V03-007 EMD0064         Ellen M. Dusseault      17-MAR-1984
0000    ;               Add support to FORMAT to recognize symbols which are subtypes
0000    ;               of a generic code.
0000    ;
0000    ;       V03-006 PRB0242         Paul Beck       28-DEC-1983 16:22
0000    ;               Add support to FORMAT for addresses ($A_)
0000    ;
0000    ;       V03-005 ROW0237         Ralph O. Weber          22-OCT-1983
0000    ;               Add TRANSLATE_ADDRESS to translate a value to an address using
0000    ;               the contents of a ADDR_TABLE or TABLE.  Add IOC$RETURN to
0000    ;               list of needed symbols.  Also correct ADD_SYMBOL to save R11;
0000    ;               it corrupts it.
0000    ;
0000    ;       V03-004 TMH0004         Tim Halvorsen           02-Aug-1983
0000    ;               Add required symbols needed for IPID/EPID conversions.
0000    ;
```

```
0000   ;       V03-003 KTA3041         Kerbey T. Altmann       17-Mar-1983
0000   ;               Add symbol for system block list head.
0000   ;
0000   ;       V03-002 KTA0103         Kerbey T. Altmann       23-Jun-1982
0000   ;               Add new symbols for lock tables and channels.
0000   ;
0000   ;       V001    TMH0001         Tim Halvorsen   05-May-1982
0000   ;               Fix bug in FORMAT which prevents a trailing $C_
0000   ;               symbol from being printed at the end of a block.
0000   ;               Display $G_ symbols in FORMAT, so that random
0000   ;               format data fields within structures can be identified.
0000   ;--
```

```
0000             .SBTTL  DECLARATIONS
0000     ;
0000     ;       SYMBOL DEFINTIONS
0000     ;
0000             $RABDEF                 ; RMS RECORD ACCESS BLOCK
0000             $STSDEF                 ; COMPLETION CODE FIELDS
0000             $DMPDEF                 ; DUMP FILE DEFINITIONS
0000             $EMBDEF <CR>            ; ERROR LOG ENTRY DEFINITIONS
0000             $OBJDEF                 ; OBJECT MODULE DEFINITIONS
0000             $SYMDEF                 ; SDA SYMBOL TABLE DEFINITIONS
0000             $NODEDEF                ; TREE NODE DEFINITIONS
0000             $TPADEF                 ; TPARSE DEFINITIONS
0000             $IRPDEF                 ; IRP DEFINITIONS (FOR BLOCK TYPE)
0000             $PHDDEF                 ; PHD DEFINITIONS (FOR INTERNAL SYMS)
0000             $DPTDEF                 ; DPT DEFINITIONS
0000             $DYNDEF                 ;
0000             $UCBDEF                 ; UNIT CONTROL BLOCK
0000
```

```
    0000               .SBTTL  SYSTEM SYMBOLS USED IN THE ANALYSIS
    0000       ;
    0000       ;       THIS DEFINES THE SYSTEM SYMBOLS AND THEIR VALUES TO BE
    0000       ;       USED LATER IN THE ANALYSIS OF THE DUMP.  EACH ENTRY IS
    0000       ;       VARIABLE LENGTH, WITH THE FIRST LONGWORD CONTAINING THE
    0000       ;       VALUE AND THE REMAINING BYTES HOLDING THE COUNTED STRING.
    0000       ;       THE TABLE IS TERMINATED BY A LONGWORD OF -1.
    0000       ;
    0000
00000000               .PSECT  ZREQSYMS,NOEXE,WRT        ; PUT AFTER DATA, BEFORE CODE
    0000
    0000               .MACRO  SYMBOL  NAME,OPTIONAL
    0000       NAME::
    0000               .IF     B,OPTIONAL
    0000               .LONG   0                 ; REQUIRED SYMBOL
    0000               .IFF
    0000               .LONG   1                 ; OPTIONAL SYMBOL (MAY NOT BE THERE)
    0000               .ENDC
    0000               .ASCIC  \NAME\
    0000               .ENDM
    0000
    0000       REQ_SYMBOLS:
    0000               SYMBOL  CLU$GL_CLUB       ; ADDRESS OF CLUSTER BLOCK (CLUB)
    0010               SYMBOL  CLU$GL_LOA_ADDR   ; BASE ADDRESS OF CLUSTRLOA CODE
    0024               SYMBOL  CTL$AL_STACK      ; PROCESS STACKS INFORMATION
    0035               SYMBOL  CTL$AL_STACKLIM   ; PROCESS STACKS LIMIT INFORMATION
    0049               SYMBOL  CTL$GL_CCBBASE    ; START OF CHANNEL CONTROL AREA
    005C               SYMBOL  CTL$GL_IMGHDRBF,OPTIONAL ; IMAGE HEADER BUFFER ADDRESS
    0070               SYMBOL  CTL$GW_CHINDX     ; HIGH WATER MARK FOR CHANNEL TABLE
    0082               SYMBOL  EXE$GB_CPUTYPE    ; PROCESSOR TYPE CODE
    0095               SYMBOL  EXE$GL_NONPAGED   ; NON-PAGED FREE LIST HEAD
    00A9               SYMBOL  EXE$GL_MP         ; BASE ADDRESS OF MP CODE
    00B7               SYMBOL  EXE$GL_PAGED      ; PAGED FREE LIST HEAD
    00C8               SYMBOL  EXE$GL_RPB        ; RESTART PARAMETER BLOCK ADDRESS
    00D7               SYMBOL  EXE$GL_SPLITADR   ; ADDRESS OF IRP LOOKASIDE POOL
    00EB               SYMBOL  EXE$GQ_SYSTIME    ; SYSTEM ABSOLUTE TIME IN NANOSECONDS
    00FE               SYMBOL  EXE$AL_STACKS     ; SYSTEM STACK INFORMATION
    0110               SYMBOL  EXE$GL_INTSTK     ; ADDRESS OF INTERRUPT STACK
    0122               SYMBOL  EXE$MCHK          ; ADDRESS OF MACHINE CHECK TRANSFER VECTOR
    012F               SYMBOL  IOC$GL_DEVLIST    ; HEAD OF DDB LIST
    0142               SYMBOL  IOC$GL_DPTLIST    ; HEAD OF DPT LIST
    0155               SYMBOL  IOC$GL_IRPCNT     ; NUMBER OF IRP ENTRIES AVAILABLE
    0167               SYMBOL  IOC$GL_IRPFL      ; IRP FREE LIST HEAD
    0178               SYMBOL  IOC$GL_LRPCNT     ; NUMBER OF LRP ENTRIES AVAILABLE
    018A               SYMBOL  IOC$GL_LRPFL      ; LRP FREE LIST HEAD
    019B               SYMBOL  IOC$GL_LRPSIZE    ; LRP PACKET SIZE
    01AE               SYMBOL  IOC$GL_LRPSPLIT   ; ADDRESS OF LRP LOOKASIDE POOL
    01C2               SYMBOL  IOC$GL_SRPCNT     ; NUMBER OF SRP ENTRIES AVAILABLE
    01D4               SYMBOL  IOC$GL_SRPFL      ; SRP FREE LIST HEAD
    01E5               SYMBOL  IOC$GL_SRPSIZE    ; SRP PACKET SIZE
    01F8               SYMBOL  IOC$GL_SRPSPLIT   ; ADDRESS OF LRP LOOKASIDE POOL
    020C               SYMBOL  IOC$RETURN        ; I/O SYSTEM RSB ROUTINE
    021B               SYMBOL  LCK$GL_HASHTBL    ; ADDRESS OF THE LOCK HASH TABLE
    022E               SYMBOL  LCK$GL_HTBLCNT    ; COUNT OF HASH TABLE ENTRIES
    0241               SYMBOL  LCK$GL_IDTBL      ; ADDRESS OF THE LOCK ID TABLE
    0252               SYMBOL  LCK$GL_MAXID      ; MAXIMUM LOCK ID
    0263               SYMBOL  MMG$AL_SYSPCB     ; ADDRESS OF SYSTEM PROCESS HEADER
```

```
0275        SYMBOL  MMG$FRSTRONLY   ; ADDRESS OF READ-ONLY EXEC CODE
0287        SYMBOL  MMG$IMGHDRBUF   ; IMAGE HEADER BUFFER (P1 SPACE)
0299        SYMBOL  MMG$GL_GPTE     ; FIRST GLOBAL PAGE TABLE ENTRY
02A9        SYMBOL  MMG$GL_MAXGPTE  ; MAXIMUM GLOBAL PAGE TABLE ENTRY
02BC        SYMBOL  MMG$GL_MAXPFN   ; MAX PFN COVERED BY PFN DATA BASE
02CE        SYMBOL  MMG$GL_RMSBASE  ; ADDRESS OF RMS CODE
02E1        SYMBOL  MMG$GL_SYSLOA_BASE      ; ADDRESS OF SYSLOA CODE
02F8        SYMBOL  MMG$GL_FPEMUL_BASE      ; ADDRESS OF FPEMUL CODE
030F        SYMBOL  MMG$GL_VAXEMUL_BASE     ; ADDRESS OF VAXEMUL CODE
0327        SYMBOL  MMG$GL_SBR      ; PHYSICAL ADDRESS OF SPT
0336        SYMBOL  MMG$GL_SPTLEN   ; LENGTH OF SPT IN PAGES
0348        SYMBOL  MMG$GL_SPTBASE  ; ADDRESS OF SYSTEM PAGE TABLE
035B        SYMBOL  MMG$GL_SYSPHD   ; ADDRESS OF SYSTEM PROCESS HEADER
036D        SYMBOL  MMG$GL_NPAGEDYN ; ADDRESS OF NON-PAGED POOL
0381        SYMBOL  MMG$GL_NPAGNEXT ; ADDRESS OF END OF NON-PAGED POOL
0395        SYMBOL  MMG$GL_PAGEDYN  ; ADDRESS OF PAGED POOL
03A8        SYMBOL  PIO$GW_IIOIMPA  ; ADDRESS OF I/O IMPURE AREA
03BB        SYMBOL  PFN$AB_STATE    ; CURRENT STATE OF PAGE FRAME
03CC        SYMBOL  PFN$AB_TYPE     ; TYPE OF PAGE FRAME ENTRY
03DC        SYMBOL  PFN$AL_BAK      ; BACKING STORE ADDRESS
03EB        SYMBOL  PFN$AL_HILIMIT  ; MAXIMUM THRESHOLD FOR PAGE LISTS
03FE        SYMBOL  PFN$AL_LOLIMIT  ; MINIMUM THRESHOLD FOR PAGE LISTS
0411        SYMBOL  PFN$AL_PTE      ; ADDRESS OF PROCESS PTE
0420        SYMBOL  PFN$AX_BLINK    ; BACKWARD PAGE LIST LINK
0431        SYMBOL  PFN$AX_FLINK    ; FORWARD PAGE LIST LINK
0442        SYMBOL  PFN$AL_HEAD     ; HEADERS FOR PAGE LISTS
0452        SYMBOL  PFN$AW_REFCNT   ; REFERENCE COUNT FOR PAGE
0464        SYMBOL  PFN$AX_WSLX     ; WORKING SET LIST INDEX
0474        SYMBOL  PFN$A_BASE      ; BASE ADDRESS OF PFN ARRAYS
0483        SYMBOL  SCH$GL_CURPCB   ; CURRENT PCB ADDRESS
0495        SYMBOL  SCH$GL_FREECNT  ; ADDRESS OF PAGE LIST COUNTS
04A8        SYMBOL  SCH$GL_PCBVEC   ; VECTOR OF PCB ADDRESSES
04BA        SYMBOL  SCH$GL_MAXPIX   ; MAXIMUM PROCESS INDEX
04CC        SYMBOL  SCS$GA_EXISTS   ; ADDRESS OF SCSLOA CODE
04DE        SYMBOL  SCS$GA_LOCALSB  ; LOCAL SYSTEM BLOCK
04F1        SYMBOL  SCS$GL_CDL      ; CONNECTION DESCRIPTOR LIST
0500        SYMBOL  SCS$GL_MSCP     ; ADDRESS OF MSCP CODE
0510        SYMBOL  SCS$GL_PDT      ; PORT DESCRIPTOR TABLE
051F        SYMBOL  SCS$GL_RDT      ; RESPONSE DESCRIPTOR TABLE
052E        SYMBOL  SCS$GQ_CONFIG   ; SYSTEM BLOCK LIST HEAD
0540        SYMBOL  SCS$GQ_DIRECT   ; SCS DIRECTORY
0552        SYMBOL  SGN$GL_PAGEDYN  ; LENGTH OF PAGED POOL
0565        SYMBOL  SGN$GW_ISPPGCT  ; INTERRUPT STACK PAGE COUNT
0578        SYMBOL  SWP$A_KSTK      ; ADDRESS OF SWAPPER STACK
0587        SYMBOL  SWP$K_KSTKSZ    ; LENGTH OF SWAPPER STACK
0598        SYMBOL  SYS$GQ_VERSION  ; VERSION OF SYSTEM (E.G. 6.01)
05AB    ;
05AB    ;   SYMBOLS USED TO EXAMINE CURRENT RUNNING SYSTEM
05AB    ;
05AB        SYMBOL  EXE$ALLOCBUF    ; ALLOCATE NON-PAGED STORAGE
05BC        SYMBOL  EXE$DEANONPAGED ; DEALLOCATE NON-PAGED STORAGE
05D0        SYMBOL  EXE$MCHK_PRTCT  ; MACHINE CHECK PROTECTION
05E3        SYMBOL  PHV$GL_PIXBAS   ; ADDRESS OF BALANCE SLOT TO PIX WORD ARRAY
05F5        SYMBOL  SCH$QAST        ; QUEUE AN AST TO ANOTHER PROCESS
0602        SYMBOL  SCH$WAKE        ; WAKE A PROCESS
060F        SYMBOL  SCH$WAITK       ; PLACE PROCESS IN WAIT QUEUE
061D        SYMBOL  SCH$GQ_SUSP     ; SUSPEND WAIT QUEUE HEADER
```

```
                                  062D      SYMBOL  SGN$GL_BALSETCT ; NUMBER OF BALANCE SET SLOTS
                                  0641      SYMBOL  SWP$GL_BSLOTSZ  ; SIZE OF EACH BALANCE SET SLOT IN PAGES
                                  0654      SYMBOL  SWP$GL_BALBASE  ; STARTING ADDRESS OF BALANCE SET SLOTS
                                  0667      SYMBOL  EXE$IPID_TO_EPID ; IPID TO EPID CONVERSION ROUTINE
                                  067C
                        0000067C  067C  REQ_SYMBOLS_LEN = . - REQ_SYMBOLS
                                  067C
      53 4D 52 00000684'010E0000' 067C  RMS:            .ASCID  'RMS'           ; STRING FOR SYMBOL "RMS"
   4B 48 43 4D 0000068F'010E0000' 0687  MCHK:           .ASCID  'MCHK'          ; STRING FOR SYMBOL "MCHK"
         50 4D 0000069B'010E0000' 0693  MP:             .ASCID  'MP'            ; STRING FOR SYMBOL "MP"
52 54 53 55 4C 43 000006A5'010E0000' 069D  CLUSTRLOA:   .ASCID  'CLUSTRLOA'     ; STRING FOR SYMBOL "CLUSTRLOA"
                        41 4F 4C  06AB
   50 43 53 4D 000006B6'010E0000' 06AE  MSCP:           .ASCID  'MSCP'          ; STRING FOR SYMBOL "MSCP"
4C 55 4D 45 50 46 000006C2'010E0000' 06BA  FPEMUL:      .ASCID  'FPEMUL'        ; STRING FOR SYMBOL "FPEMUL"
55 4D 45 58 41 56 000006D0'010E0000' 06C8  VAXEMUL:     .ASCID  'VAXEMUL'       ; STRING FOR SYMBOL "VAXEMUL"
                              4C  06D6
41 4F 4C 53 59 53 000006DF'010E0000' 06D7  SYSLOA:      .ASCID  'SYSLOA'        ; STRING FOR SYMBOL "SYSLOA"
41 4F 4C 53 43 53 000006ED'010E0000' 06E5  SCSLOA:      .ASCID  'SCSLOA'        ; STRING FOR SYMBOL "SCSLOA"
```

```
                                 06F3        .SBTTL   UCB DEVICE EXTENSIONS
                                 06F3
                                 06F3   DEVCLASS_TABLE:
                                 06F3       ADDR_TABLE      DC$_,<-
                                 06F3               <DISK,UCB_DT>,-            ; disk and tape routine to check validity
                                 06F3               <TAPE,UCB_DT>,-            ; disk and tape
                                 06F3               <TERM,UCB_TERM>,-          ; routine for terminal devices
                                 06F3               <BUS,UCB_BUS>,-            ; routine for network devices
                                 06F3               <MAILBOX,UCB_MAIL>,-       ; routine for mailbox devices
                                 06F3               <JOURNAL,UCB_JOURNAL>-     ; routine for journal devices
                                 06F3               >
                                 072B
                                 072B   DEVICE_ROUT:
                       00000000  072B               .LONG 0                    ; will contain address of device
                                 072F                                          ; dependent routine
                                 072F   ;
                                 072F   ; Define the device extensions of the UCB
                                 072F   ;
   4C 4E 4A 00000737'010E0000'   072F   JOURNAL_EXT:    .ASCID   'JNL'         ; All journal extensions be ucb$x_jnl_xxx
                       00000000  073A                   .LONG    0             ; Denotes end of table
      42 4D 00000746'010E0000'   073E   MAILBX_EXT:     .ASCID   'MB'          ; All mailbox extensions be ucb$x_mb_xxxx
                       00000000  0748                   .LONG    0             ;
      54 4E 00000754'010E0000'   074C   NETMBX_EXT:     .ASCID   'NT'          ; Network Mailbox
                       00000000  0756                   .LONG    0
      54 54 00000762'010E0000'   075A   LTERM_EXT:      .ASCID   'TT'          ; Local terminal extension prefixes
      4C 54 0000076C'010E0000'   0764                   .ASCID   'TL'          ; TL - Logical Terminal
      50 54 00000776'010E0000'   076E                   .ASCID   'TP'          ; Terminal Port Driver Dependent
                       00000000  0778                   .LONG    0
      4C 54 00000784'010E0000'   077C   RTERM_EXT:      .ASCID   'TL'          ; TL - Logical Terminal
   54 54 52 0000078E'010E0000'   0786                   .ASCID   'RTT'         ; Remote Terminal
      54 43 00000799'010E0000'   0791                   .ASCID   'CT'          ; CTERM Driver Only
                       00000000  079B                   .LONG    0
      49 4E 000007A7'010E0000'   079F   BUS_EXT:        .ASCID   'NI'          ; NI Device
                       00000000  07A9                   .LONG    0
                                 07AD   ;
                                 07AD   ; The device extensions for disks,tapes, and error logging do not have a common
                                 07AD   ; prefix for all its symbols.  Thus we are forced to list each symbol and do
                                 07AD   ; a compare of the current symbol against all.  I know it is brute force but I
                                 07AD   ; can not think of any other way.  If a symbol for these devices gets added
                                 07AD   ; to the extensions in $ucbdef, it will have to be added here also if it is
                                 07AD   ; to be displayed by format.
                                 07AD   ;
                                 07AD   Errorlog_ext:
   45 56 41 4C 53 000007B5'010E0000'   07AD         .ASCID   'SLAVE'
         52 50 53 000007C2'010E0000'   07BA         .ASCID   'SPR'
         58 45 46 000007CD'010E0000'   07C5         .ASCID   'FEX'
         58 45 43 000007D8'010E0000'   07D0         .ASCID   'CEX'
         42 4D 45 000007E3'010E0000'   07DB         .ASCID   'EMB'
      43 4E 55 46 000007EE'010E0000'   07E6         .ASCID   'FUNC'
         43 50 44 000007FA'010E0000'   07F2         .ASCID   'DPC'
45 4C 5F 4C 52 45 00000805'010E0000'   07FD         .ASCID   'ERL_LENGTH'
                        48 54 47 4E    080B
                           00000000    080F         .LONG    0
                                       0813
                                       0813   Disks_tapes:
42 44 44 5F 50 32 0000081B'010E0000'   0813         .ASCID   '2P_DDB'
4E 49 4C 5F 50 32 00000829'010E0000'   0821         .ASCID   '2P_LINK'
```

```
                                   4B  082F
54 4C 41 5F 50 32 00000838'010E0000'  0830   305             .ASCID  '2P_ALTUCB'
                             42 43 55  083E
51 45 53 52 49 44 00000849'010E0000'  0841   306             .ASCID  'DIRSEQ'
54 4E 43 4C 4E 4F 00000857'010E0000'  084F   307             .ASCID  'ONLCNT'
4F 4C 42 58 41 4D 00000865'010E0000'  085D   308             .ASCID  'MAXBLOCK'
                                4B 43  086B
44 52 4F 43 45 52 00000875'010E0000'  086D   309             .ASCID  'RECORD'
4E 43 42 58 41 4D 00000883'010E0000'  087B   310             .ASCID  'MAXBCNT'
                                   54  0889
         42 43 43 44 00000892'010E0000'  088A   311             .ASCID  'DCCB'
                             00000000  0896   312             .LONG   0
                                       089A   313
                                       089A   314 mscp_ext:
         42 44 44 43 000008A2'010E0000'  089A   315             .ASCID  'CDDB'
44 44 43 5F 50 32 000008AE'010E0000'  08A6   316             .ASCID  '2P_CDDB'
                                   42  08B4
4C 5F 42 44 44 43 000008BD'010E0000'  08B5   317             .ASCID  'CDDB_LINK'
                             4B 4E 49  08C3
            54 44 43 000008CE'010E0000'  08C6   318             .ASCID  'CDT'
49 5F 54 49 4E 55 000008D9'010E0000'  08D1   319             .ASCID  'UNIT_ID'
                                   44  08DF
4E 55 50 43 53 4D 000008E8'010E0000'  08E0   320             .ASCID  'MSCPUNIT'
                                54 49  08EE
46 5F 54 49 4E 55 000008F8'010E0000'  08F0   321             .ASCID  'UNIT_FLAGS'
                          53 47 41 4C  08FE
45 44 50 43 53 4D 0000090A'010E0000'  0902   322             .ASCID  'MSCPDEVPARAM'
                    4D 41 52 41 50 56  0910
43 5F 54 49 41 57 0000091E'010E0000'  0916   323             .ASCID  'WAIT_CDDB'
                             42 44 44  0924
                             00000000  0927   324             .LONG   0
                                       092B   325
                                       092B   326 LDISK_EXT:
   41 49 44 45 4D 00000933'010E0000'  092B   327             .ASCID  'MEDIA'
            52 43 42 00000940'010E0000'  0938   328             .ASCID  'BCR'
            31 43 45 0000094B'010E0000'  0943   329             .ASCID  'EC1'
            32 43 45 00000956'010E0000'  094E   330             .ASCID  'EC2'
54 45 53 46 46 4F 00000961'010E0000'  0959   331             .ASCID  'OFFSET'
58 44 4E 46 46 4F 0000096F'010E0000'  0967   332             .ASCID  'OFFNDX'
43 54 52 46 46 4F 0000097D'010E0000'  0975   333             .ASCID  'OFFRTC'
46 55 42 5F 58 44 0000098B'010E0000'  0983   334             .ASCID  'DX_BUF'
50 46 42 5F 58 44 00000999'010E0000'  0991   335             .ASCID  'DX_BFPNT'
                                54 4E  099F
44 58 52 5F 58 44 000009A9'010E0000'  09A1   336             .ASCID  'DX_RXDB'
                                   42  09AF
52 43 42 5F 58 44 000009B8'010E0000'  09B0   337             .ASCID  'DX_BCR'
54 43 53 5F 58 44 000009C6'010E0000'  09BE   338             .ASCID  'DX_SCTCNT'
                             54 4E 43  09CC
                             00000000  09CF   339             .LONG   0
                                       09D3   340
```

```
             09D3          .SBTTL  STORAGE DEFINTIONS
             09D3  ;
             09D3  ;       STORAGE DEFINITIONS
             09D3  ;
             09D3
         00000000          .PSECT  SDADATA,NOEXE,WRT
             0000
             0000  STB_PTR:
00000004     0000          .BLKL   1                       ; POINTER FOR SCANNING STB FILE
             0004
             0004  TRANSLTH:
00000008     0004          .BLKL   1                       ; RESULT DESCRIPTOR FOR
0000000C     0008          .BLKL   1                       ; "SYMBOLIZE" ROUTINE
             000C
             000C  SYMFL:                                  ; SYMBOL VALUE TREE ROOT
00000010     000C          .BLKL   1
             0010  ALPFL:                                  ; ALPHABETIC TREE ROOT
00000014     0010          .BLKL   1
             0014  NSYMBOLS:                               ; NUMBER OF SYMBOLS IN TABLE
00000018     0014          .BLKL   1
             0018
             0018  OFFSET_TABLE:
00000020     0018          .BLKL   2                       ; STARTING/ENDING ADDRESSES
             0020
             0020  STRUCTURE::
00000028     0020          .BLKL   2                       ; DESCRIPTOR OF STRUCTURE NAME
             0028
             0028  UCB_STRUCT:
000000B8     0028          .BLKB   UCB$C_LENGTH            ; UNIT CONTROL BLOCK
             00B8
         00000000          .PSECT  SYMBOLS,EXE,NOWRT
             0000
             0000          .DEFAULT DISPLACEMENT,LONG
```

```
                  0000             .SBTTL  READ-ONLY DATA DEFINITIONS
                  0000
                  0000      ;
                  0000      ;      READ-ONLY DATA DEFINITIONS
                  0000      ;
                  0000
                  0000             .MACRO  DEFINE  REG,VALUE,?L1
                  0000             .SAVE
                  0000             .PSECT  LITERALS,EXE,NOWRT
                  0000      L1:    STRING  <REG>
                  0000             .RESTORE
                  0000             .LONG   L1
                  0000             .IF     B,<VALUE>
                  0000             .LONG   ^XC0000000+PHD$L_'REG'
                  0000             .IFF
                  0000             .LONG   VALUE
                  0000             .ENDC
                  0000             .ENDM
                  0000
                  0000      REG_TABLE:
                  0000             DEFINE  R0
                  0008             DEFINE  R1
                  0010             DEFINE  R2
                  0018             DEFINE  R3
                  0020             DEFINE  R4
                  0028             DEFINE  R5
                  0030             DEFINE  R6
                  0038             DEFINE  R7
                  0040             DEFINE  R8
                  0048             DEFINE  R9
                  0050             DEFINE  R10
                  0058             DEFINE  R11
                  0060             DEFINE  R12
                  0068             DEFINE  R13
                  0070             DEFINE  AP,<^XC0000000+PHD$L_R12>
                  0078             DEFINE  FP,<^XC0000000+PHD$L_R13>
                  0080             DEFINE  USP
                  0088             DEFINE  SSP
                  0090             DEFINE  ESP
                  0098             DEFINE  KSP
                  00A0             DEFINE  PC
                  00A8             DEFINE  PSL
                  00B0             DEFINE  P0BR
                  00B8             DEFINE  P0LR,<^XC0000000+PHD$L_P0LRASTL>
                  00C0             DEFINE  P1BR
                  00C8             DEFINE  P1LR
                  00D0             DEFINE  G,<^X80000000>
                  00D8             DEFINE  H,<^X7FFE0000>
00000000 00000000 00E0             .QUAD   0                       ; END OF TABLE
                  00E8
                  00E8      PRTCTL1:
                  00E8             STRING  <!15AC !XL! >
                  00FB      PRTCTL2:
                  00FB             STRING  ^/!15AC !XL => !XL/
                  0113
                  0113      TRANSCTL1:
                  0113             STRING  <!AC+!3XL>              ; SYMBOL+OFFSET
```

```
                        0123   433
                        0123   434 TRANSCTL2:
                        0123   435         STRING  <!AC>                   ; SYMBOL (IF 0 OFFSET)
                        012E   436
                        012E   437 LENGTH_SYM:
  48 54 47 4E 45 4C 5F  012E   438         .ASCII /_LENGTH/                ; DENOTES THE ONLY CONSTANT
              00000007  0135   439         L_LEN=.-LENGTH_SYM              ; SYMBOL TO BE DISPLAYED WHEN
                        0135   440                                         ; FORMATTING A DATA STRUCTURE
                        0135   441 UCB_PREFIX:
              42 43 55  0135   442         .ASCII /UCB/                    ; PREFIX IS UCB
              00000003  0138   443         PREFIX_LEN=.-UCB_PREFIX
                        0138   444
                        0138   445
```

```
                         0138          .SBTTL  REWIND_STB - REWIND SYSTEM SYMBOL TABLE FILE
                         0138  ;---
                         0138  ;
                         0138  ;       REWIND_STB
                         0138  ;
                         0138  ;       THIS ROUTINE INITIALIZES THINGS SO THAT ANOTHER PASS
                         0138  ;       CAN BE MADE OVER THE SYSTEM SYMBOL TABLE FILE.
                         0138  ;
                         0138  ;   INPUTS:
                         0138  ;
                         0138  ;       NONE
                         0138  ;
                         0138  ;   OUTPUTS:
                         0138  ;
                         0138  ;       THE FILE IS REWOUND.
                         0138  ;       STB_PTR = 0 TO INDICATE NO RECORDS READ YET.
                         0138  ;
                         0138  ;---
                         0138
                         0138  REWIND_STB::
                   0000  0138          .WORD   0
                         013A
                         013A  ;       $REWIND STB                     ; REWIND ACTUAL FILE
                         013A  ;       SIGNAL  RMS,STB
      00000000'EF   D4   013A          CLRL    STB_PTR                 ; INITIALIZE SCAN POINTER
                    04   0140          RET
```

```
                         0141          .SBTTL  GET_SYMBOL - GET NEXT GSD SYMBOL ENTRY FROM STB
                         0141  ;---
                         0141  ;
                         0141  ;       GET_SYMBOL
                         0141  ;
                         0141  ;       THIS ROUTINE GETS THE NEXT GSD-SYMBOL DEFINTION
                         0141  ;       ENTRY FROM THE SYSTEM SYMBOL TABLE FILE.
                         0141  ;
                         0141  ; INPUTS:
                         0141  ;
                         0141  ;       STB_PTR = POINTER TO THE NEXT GSD ENTRY IN THE RECORD
                         0141  ;                 OR ZERO IF THE FILE IS REWOUND.
                         0141  ;
                         0141  ; OUTPUTS:
                         0141  ;
                         0141  ;       R0 = FALSE IF END OF FILE IS REACHED, ELSE TRUE
                         0141  ;       R1 = POINTER TO THE SYMBOL DEFINITION ENTRY
                         0141  ;
                         0141  ;---
                         0141
                         0141  GET_SYMBOL::
                  003C   0141          .WORD   ^M<R2,R3,R4,R5>
                         0143
53   00000000'EF   DE    0143          MOVAL   STB,R3
        52  22 A3   3C    014A          MOVZWL  RAB$W_RSZ(R3),R2          ; GET LENGTH OF RECORD
      52   0000'C2  9E    014E          MOVAB   STB_BUFFER(R2),R2         ; ENDING ADDRESS OF RECORD
51   00000000'EF   D0    0153          MOVL    STB_PTR,R1                ; GET BUFFER POINTER
              3D   12    015A          BNEQ    20$                       ; BRANCH IF VALID
                         015C  10$:
                         015C          $GET    (R3)                      ; GET NEXT STB RECORD
00000000'8F   50   D1    0165          CMPL    R0,#RMS$_EOF              ; CHECK FOR END OF FILE
              03   12    016C          BNEQ    15$                       ; IF NOT, GO ON
              50   D4    016E          CLRL    R0                        ; INDICATE FAILURE
                   04    0170          RET
                         0171  15$:    SIGNAL  RMS,(R3)                  ; CHECK FOR RMS ERROR
51   00000000'EF   9E    0184          MOVAB   STB_BUFFER,R1             ; INITIALIZE POINTER
          01   61   91    018B          CMPB    OBJ$B_TYPE(R1),#OBJ$C_GSD  ; ONLY LOOK AT GSD RECORDS
              CC   12    018E          BNEQ    10$                       ; BRANCH IF NOT GSD RECORD
     52   22 A3   3C    0190          MOVZWL  RAB$W_RSZ(R3),R2          ; GET LENGTH OF RECORD
         52   51   C0    0194          ADDL    R1,R2                     ; ENDING ADDRESS OF RECORD
              51   D6    0197          INCL    R1                        ; SKIP RECORD IDENTIFICATION
                         0199  20$:
         52   51   D1    0199          CMPL    R1,R2                     ; CHECK IF END OF RECORD
              BE   1E    019C          BGEQU   10$                       ; BRANCH IF SO
         50   61   9A    019E          MOVZBL  OBJ$B_GSD_TYPE(R1),R0     ; GET GSD ENTRY TYPE CODE
                         01A1          ASSUME  OBJ$C_GSD_PSC EQ 0
                         01A1          ASSUME  OBJ$C_GSD_SYM EQ 1
                         01A1          ASSUME  OBJ$C_GSD_EPM EQ 2
                         01A1          ASSUME  OBJ$C_GSD_PRO EQ 3
                         01A1          CASE    R0,TYPE=B,-               ; CASE ON GSD ENTRY TYPE CODE
                         01A1                  <30$,25$,40$,10$>
              50   DD    01AD          PUSHL   R0                        ; OUTPUT UNKNOWN TYPE
                         01AF          SIGNAL  1,BADGSD                  ; UNKNOWN TYPE OF GSD ENTRY
         99   11         01C1          BRB     10$                       ; SKIP TO NEXT RECORD
                         01C3  ;
                         01C3  ;       OBJ$C_GSD_SYM - GLOBAL SYMBOL DEFINITION
                         01C3  ;
```

```
      15 02 A1    01    E1  01C3   531 25$:   BBC     #OBJ$V_SYM_DEF,OBJ$W_SYM_FLAGS(R1),27$ ; BRANCH IF REFERENCE
         50    09 A1    9A  01C8   532        MOVZBL  OBJ$T_SYM_NAME(R1),R0    ; GET LENGTH OF STRING
00000000'EF    0A A140  9E  01CC   533        MOVAB   OBJ$T_SYM_NAME+1(R1)[R0],STB_PTR ; SKIP PAST ENTRY
                            01D5   534        STATUS  SUCCESS
                  04        01DC   535        RET
                            01DD   536 ;
                            01DD   537 ;      OBJ$C_GSD_SYM - GLOBAL SYMBOL REFERENCE
                            01DD   538 ;
         50    04 A1    9A  01DD   539 27$:   MOVZBL  OBJ$T_SYM_NAME-5(R1),R0 ; LENGTH OF STRING
      51       05 A140  9E  01E1   540        MOVAB   OBJ$T_SYM_NAME-5+1(R1)[R0],R1 ; SKIP PAST ENTRY
                  B1    11  01E6   541        BRB     20$                     ; CONTINUE SEARCHING
                            01E8   542 ;
                            01E8   543 ;      OBJ$C_GSD_PSC - PSECT DEFINITION
                            01E8   544 ;
         50    08 A1    9A  01E8   545 30$:   MOVZBL  OBJ$T_PSC_NAME(R1),R0           ; GET LENGTH OF STRING
      51       09 A140  9E  01EC   546        MOVAB   OBJ$T_PSC_NAME+1(R1)[R0],R1     ; SKIP THIS ENTRY
                  A6    11  01F1   547        BRB     20$                     ; CONTINUE SEARCHING
                            01F3   548 ;
                            01F3   549 ;      OBJ$C_GSD_EPM - ENTRY POINT AND MASK DEFINITION
                            01F3   550 ;
         50    0B A1    9A  01F3   551 40$:   MOVZBL  OBJ$T_EPM_NAME(R1),R0   ; LENGTH OF NAME
00000000'EF    0C A140  9E  01F7   552        MOVAB   OBJ$T_EPM_NAME+1(R1)[R0],STB_PTR  ; SAVE POINTER
                            0200   553 ;      THE FOLLOWING MAKES THE EPM ENTRY LOOK LIKE A SYM ENTRY.
                  51    DD  0200   554        PUSHL   R1                      ; SAVE POINTER TO ENTRY
                  50    D6  0202   555        INCL    R0                      ; LENGTH OF COUNTED STRING
   09 A1    0B A1 50    28  0204   556        MOVC    R0,OBJ$T_EPM_NAME(R1),OBJ$T_SYM_NAME(R1)
         51       8E    D0  020A   557        MOVL    (SP)+,R1                ; RESTORE POINTER TO ENTRY
                            020D   558        STATUS  SUCCESS
                  04        0214   559        RET
```

```
                                  0215          .SBTTL  ADD_SYMBOL -- ADD SYMBOL TO SYMBOL TABLE
                                  0215     ;---
                                  0215     ;
                                  0215     ;    ADD_SYMBOL
                                  0215     ;
                                  0215     ;    THIS ROUTINE ADDS A SYMBOL TO THE LOCAL SYMBOL TABLE.
                                  0215     ;
                                  0215     ; INPUTS:
                                  0215     ;
                                  0215     ;    4(AP) = LENGTH OF SYMBOL
                                  0215     ;    8(AP) = ADDRESS OF SYMBOL
                                  0215     ;    12(AP) = SYMBOL VALUE
                                  0215     ;    RELOCATE_BASE = SYMBOL RELOCATION BASE (NON-ZERO ON READ/RELOCATE)
                                  0215     ;                    (ONLY USED ON NEW SYMBOLS, NOT ON SYMBOL REPLACEMENTS)
                                  0215     ;
                                  0215     ;---
                                  0215
                             0FFC 0215          .ENTRY  ADD_SYMBOL,-
                                  0217                  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                  0217
                                  0217     ;
                                  0217     ;    TRUNCATE SYMBOL TO 31 CHARACTERS
                                  0217     ;
           1F    04 AC       B1   0217          CMPW    4(AP),#31                 ; 31 CHARACTERS?
                    04       15   021B          BLEQ    5$                        ; BRANCH IF NO PROBLEM
              04 AC 1F       B0   021D          MOVW    #31,4(AP)                 ; TRUNCATE TO 31 CHARACTERS
                                  0221     5$:
                                  0221     ;
                                  0221     ;    SEARCH SYMBOL TABLE TO SEE IF ALREADY EXISTS
                                  0221     ;
      57    00000010'EF      D0   0221          MOVL    ALPFL,R7                  ; ROOT OF ALPHA SORTED TREE
                    26       13   0228     10$: BEQL    50$                       ; BRANCH IF NOT IN SYMBOL TABLE
           56    0A A7       D0   022A          MOVL    NODE$L_PTR(R7),R6         ; ADDRESS OF SYMBOL TABLE ENTRY
           50    04 A6       9A   022E          MOVZBL  SYM$T_SYMBOL(R6),R0       ; LENGTH OF SYMBOL
  50    20    08 BC 04 AC    2D   0232          CMPC5   4(AP),@8(AP),#^A' ',R0,SYM$T_SYMBOL+1(R6)
                 05 A6            0239
                    0D       13   023B          BEQL    40$                       ; BRANCH IF FOUND
                    05       1A   023D          BGTRU   20$                       ; BRANCH IF GREATER
              57    67       D0   023F          MOVL    NODE$L_LEFT(R7),R7        ; GO TO LEFT BRANCH OF TREE
                    E4       11   0242          BRB     10$                       ; CONTINUE DOWN TREE
           57    04 A7       D0   0244     20$: MOVL    NODE$L_RIGHT(R7),R7       ; GO TO RIGHT SIDE OF TREE
                    DE       11   0248          BRB     10$                       ; CONTINUE DOWN TREE
                                  024A     ;
                                  024A     ;    SYMBOL ALREADY EXISTS, SIMPLY RE-DEFINE VALUE
                                  024A     ;
           66    0C AC       D0   024A     40$: MOVL    12(AP),SYM$L_VALUE(R6)    ; RE-DEFINE SYMBOL
                    20       11   024E          BRB     90$                       ; AND EXIT
                                  0250     ;
                                  0250     ;    CREATE NEW SYMBOL TABLE ENTRY (RELOCATING VALUE, IF NECESSARY)
                                  0250     ;
           52    04 AC       7D   0250     50$: MOVQ    4(AP),R2                  ; GET DESCRIPTOR OF SYMBOL NAME
           54    0C AC       D0   0254          MOVL    12(AP),R4                 ; AND GET VALUE OF SYMBOL
              00000000'EF    D5   0258          TSTL    RELOCATE_BASE             ; ARE WE RELOCATING?
                    0E       13   025E          BEQL    15$                       ; BRANCH IF NOT
      50    54    F0 8F      78   0260          ASHL    #-16,R4,R0                ; GREATER THAN 16k?
                    07       12   0265          BNEQ    15$                       ; ONLY RELOCATE PROGRAM OFFSETS
      54    00000000'EF      C0   0267          ADDL    RELOCATE_BASE,R4          ; RELOCATE THE SYMBOL
```

```
08  10  026E   617 15$:   BSBB    ADD_NEW_ENTRY        ; ADD NEW SYMBOL TABLE ENTRY
        0270   618
        0270   619 90$:   STATUS  SUCCESS              ; RETURN SUCCESS
    04  0277   620        RET
```

```
                              0278        .SBTTL  ADD_NEW_ENTRY, ADD NEW ENTRY TO TABLE
                              0278  ;---
                              0278  ;
                              0278  ;       INSERT A NEW SYMBOL TABLE ENTRY INTO THE SYMBOL TABLE
                              0278  ;
                              0278  ;  INPUTS:
                              0278  ;
                              0278  ;       R2/R3 = DESCRIPTOR OF SYMBOL
                              0278  ;       R4 = SYMBOL VALUE
                              0278  ;
                              0278  ; OUTPUTS:
                              0278  ;
                              0278  ;       SYMBOL ENTRY INSERTED INTO CORRECT POSITION
                              0278  ;
                              0278  ;       R2-R5, R8-R11 DESTROYED.
                              0278  ;---
                              0278
                              0278  ADD_NEW_ENTRY:
                   24   DD    0278        PUSHL   #SYM$C_LENGTH           ; LENGTH OF BLOCK
      000003D4'EF  01   FB    027A        CALLS   #1,ALLOCATE             ; ALLOCATE A SYMBOL TABLE ENTRY
              58   51   D0    0281        MOVL    R1,R8                   ; SET ADDRESS OF NEW ENTRY
              68   54   D0    0284        MOVL    R4,SYM$L_VALUE(R8)      ; STORE VALUE
           04 A8   52   90    0287        MOVB    R2,SYM$T_SYMBOL(R8)     ; SET SYMBOL LENGTH
      05 A8   63   52   28    028B        MOVC3   R2,(R3),SYM$T_SYMBOL+1(R8)
        00000014'EF     D6    0290        INCL    NSYMBOLS                ; INCREMENT NUMBER OF SYMBOLS
                              0296  ;
                              0296  ;       INSERT ENTRY INTO ASCENDING VALUE ORDERED TREE
                              0296  ;
      59   0000000C'EF  D0    0296        MOVL    SYMFL,R9                ; INITIALIZE AT ROOT OF TREE
          5B     BF'AF  9E    029D        MOVAB   B^COMPARE_VALUE,R11     ; ADDRESS OF COMPARE ROUTINE
                   40   10    02A1        BSBB    INSERT                  ; INSERT INTO VALUE TREE
      0000000C'EF  59   D0    02A3        MOVL    R9,SYMFL                ; SET NEW ROOT OF TREE
                              02AA  ;
                              02AA  ;       INSERT ENTRY INTO ASCENDING ALPHABETICALLY ORDERED TREE
                              02AA  ;
      59   00000010'EF  D0    02AA        MOVL    ALPFL,R9                ; INITIALIZE AT ROOT OF TREE
          5B     C7'AF  9E    02B1        MOVAB   B^COMPARE_ALPHA,R11     ; ADDRESS OF COMPARE ROUTINE
                   2C   10    02B5        BSBB    INSERT                  ; INSERT INTO ALPHA TREE
      00000010'EF  59   D0    02B7        MOVL    R9,ALPFL                ; SET NEW ROOT OF TREE
                        05    02BE        RSB
```

```
                  02BF          .SBTTL  COMPARE_VALUE, COMPARE SYMBOL ENTRIES BY VALUE
                  02BF     ;---
                  02BF     ;
                  02BF     ;       COMPARE THE VALUES OF 2 ENTRIES
                  02BF     ;
                  02BF     ; INPUTS:
                  02BF     ;
                  02BF     ;       R8 = ADDRESS OF SYMBOL TABLE ENTRY TO BE INSERTED
                  02BF     ;       R9 = ADDRESS OF CURRENT NODE
                  02BF     ;
                  02BF     ; OUTPUTS:
                  02BF     ;
                  02BF     ;       PSL CONDITIONS SET.
                  02BF     ;---
                  02BF
                  02BF     COMPARE_VALUE:
52   0A A9   D0   02BF             MOVL    NODE$L_PTR(R9),R2        ; ADDRESS OF SYMBOL TABLE ENTRY
   62   68   D1   02C3             CMPL    SYM$L_VALUE(R8),SYM$L_VALUE(R2)
             05   02C6             RSB
```

```
                         02C7            .SBTTL  COMPARE_ALPHA, COMPARE SYMBOL ENTRIES BY NAME
                         02C7      ;---
                         02C7      ;
                         02C7      ;       COMPARE THE STRING NAMES OF 2 ENTRIES
                         02C7      ;
                         02C7      ; INPUTS:
                         02C7      ;
                         02C7      ;       R8 = ADDRESS OF SYMBOL TABLE ENTRY TO BE INSERTED
                         02C7      ;       R9 = ADDRESS OF CURRENT NODE
                         02C7      ;
                         02C7      ; OUTPUTS:
                         02C7      ;
                         02C7      ;       PSL CONDITIONS SET.
                         02C7      ;---
                         02C7
                         02C7      COMPARE_ALPHA:
         52    0A A9  D0 02C7            MOVL    NODE$L_PTR(R9),R2             ; ADDRESS OF SYMBOL TABLE ENTRY
   05 A2       05 A8  91 02CB            CMPB    SYM$T_SYMBOL+1(R8),SYM$T_SYMBOL+1(R2)    ; CHECK 1ST CHAR
                  10  12 02D0            BNEQ    90$                           ; BRANCH IF THATS ENOUGH
         50    04 A8  9A 02D2            MOVZBL  SYM$T_SYMBOL(R8),R0           ; GET LENGTH OF NEW NAME
         51    04 A2  9A 02D6            MOVZBL  SYM$T_SYMBOL(R2),R1           ; GET LENGTH OF CURRENT NAME
20    05 A8       50  2D 02DA            CMPC5   R0,SYM$T_SYMBOL+1(R8),#^A' ',-   ; COMPARE STRINGS
      05 A2       51     02DF                    R1,SYM$T_SYMBOL+1(R2)
                      05 02E2      90$:  RSB
```

```
                     02E3          .SBTTL  INSERT, RECURSIVE ROUTINE TO INSERT INTO TREE
                     02E3   ;---
                     02E3   ;
                     02E3   ;       INSERT
                     02E3   ;
                     02E3   ;       RECURSIVE SUBROUTINE TO INSERT THE NEW NODE INTO THE TREE
                     02E3   ;
                     02E3   ; INPUTS:
                     02E3   ;
                     02E3   ;       R8  = POINTER TO NEW SYMBOL TABLE ENTRY
                     02E3   ;       R9  = POINTER TO CURRENT NODE IN TREE
                     02E3   ;       R11 = ADDRESS OF ROUTINE TO COMPARE NEW AND CURRENT KEYS
                     02E3   ;
                     02E3   ; OUTPUTS:
                     02E3   ;
                     02E3   ;       R9 = NEW POINTER TO CURRENT NODE IN TREE
                     02E3   ;
                     02E3   ;---
                     02E3
            59   D5  02E3   INSERT: TSTL    R9                      ; NULL POINTER? (BOTTOM OF TREE)
            19   12  02E5           BNEQ    10$                     ; BRANCH IF NOT
                     02E7   ;
                     02E7   ;       REACHED BOTTOM OF TREE - INSERT INTO TREE AT BOTTOM
                     02E7   ;
            0E   DD  02E7           PUSHL   #NODE$C_LENGTH          ; LENGTH OF NODE
000003D4'EF 01   FB  02E9           CALLS   #1,ALLOCATE             ; ALLOCATE A NEW NODE
        59  51   D0  02F0           MOVL    R1,R9                   ; SET ADDRESS OF NEW NODE
     0A A9  58   D0  02F3           MOVL    R8,NODE$L_PTR(R9)       ; SET ADDR OF SYMBOL ENTRY
            69   7C  02F7           CLRQ    NODE$L_LEFT(R9)         ; ZERO LEFT AND RIGHT POINTERS
         08 A9   B4  02F9           CLRW    NODE$W_BAL(R9)          ; SET BALANCE TO ZERO
        50  01   D0  02FC           MOVL    #1,R0                   ; INDICATE NOT BALANCED
            05       02FF           RSB
                     0300   ;
                     0300   ;       CHECK IF KEY IS TO THE LEFT OR RIGHT SUBTREE
                     0300   ;
            6B   16  0300   10$:    JSB     (R11)                   ; COMPARE KEY ON CURRENT NODE
            67   1A  0302           BGTRU   200$                    ; BRANCH IF TO THE RIGHT SIDE
                     0304           BEQL    300$                    ; BRANCH IF ALREADY EXISTS
                     0304   ;
                     0304   ;       INSERT THE NODE INTO THE LEFT SUBTREE
                     0304   ;
            59   DD  0304           PUSHL   R9                      ; SAVE POINTER OF FATHER
        59  69   D0  0306           MOVL    NODE$L_LEFT(R9),R9      ; SETUP POINTER TO LEFT SON
            D8   10  0309           BSBB    INSERT                  ; INSERT INTO LEFT SUBTREE
        51  59   D0  030B           MOVL    R9,R1                   ; SAVE NEW ADDRESS OF SUBTREE
            59 8ED0  030E           POPL    R9                      ; RETURN TO FATHER NODE
        69  51   D0  0311           MOVL    R1,NODE$L_LEFT(R9)      ; POINT TO NEW LEFT SUBTREE
     06  50      E8  0314           BLBS    R0,110$                 ; BRANCH IF NOT BALANCED
       00B9      31  0317   40$:    BRW     90$                     ; EXIT
       00B4      31  031A   50$:    BRW     80$                     ; EXIT - MARK IN BALANCE
                     031D   ;
                     031D   ;       THE LEFT SUBTREE HAS GROWN HIGHER - RESTORE BALANCE
                     031D   ;
         08 A9   B7  031D   110$:   DECW    NODE$W_BAL(R9)          ; PERFORM LEFT SHIFT OF TREE
            F8   13  0320           BEQL    50$                     ; EXIT IF IN PERFECT BALANCE
      F1 08 A9   E8  0322           BLBS    NODE$W_BAL(R9),40$      ; EXIT IF AVL BLANACED
        51  69   D0  0326           MOVL    NODE$L_LEFT(R9),R1      ; R1 = POINTER TO LEFT SUBTREE
         08 A1   B5  0329           TSTW    NODE$W_BAL(R1)          ; TEST BALANCE ON THAT SIDE
```

```
        11    18  032C          BGEQ    120$                    ; BRANCH IF NEED DOUBLE ROTATION
                  032E   ;      PERFORM SINGLE LL ROTATION
  69   04 A1  D0  032E          MOVL    NODE$L_RIGHT(R1),NODE$L_LEFT(R9)
  04 A1   59  D0  0332          MOVL    R9,NODE$L_RIGHT(R1)
     08 A9    B4  0336          CLRW    NODE$W_BAL(R9)          ; MARK IN BALANCE
    59    51  D0  0339          MOVL    R1,R9                   ; SET NEW ROOT OF SUBTREE
       008F   31  033C          BRW     250$                    ; INDICATE IN PERFECT BALANCE
                  033F   ;      PERFORM DOUBLE LR ROTATION
  52   04 A1  D0  033F   120$:  MOVL    NODE$L_RIGHT(R1),R2     ; R2 = RIGHT SON OF LEFT SUBTREE
  04 A1   62  D0  0343          MOVL    NODE$L_LEFT(R2),NODE$L_RIGHT(R1)
    62    51  D0  0347          MOVL    R1,NODE$L_LEFT(R2)
  69   04 A2  D0  034A          MOVL    NODE$L_RIGHT(R2),NODE$L_LEFT(R9)
  04 A2   59  D0  034E          MOVL    R9,NODE$L_RIGHT(R2)
     08 A9    B4  0352          CLRW    NODE$W_BAL(R9)          ; INITIALIZE BALANCE INDICATORS
     08 A1    B4  0355          CLRW    NODE$W_BAL(R1)
     08 A2    B5  0358          TSTW    NODE$W_BAL(R2)          ; CHECK BALANCE OF NEW SUBTREE
        6E    13  035B          BEQL    240$                    ; SKIP IF IN BALANCE
        06    14  035D          BGTR    130$                    ; BRANCH IF LEFT SIDE HEAVY
  08 A9   01  B0  035F          MOVW    #1,NODE$W_BAL(R9)       ; MARK RIGHT SIDE HEAVY
        66    11  0363          BRB     240$                    ; SET NEW SUBTREE AND EXIT
  08 A1   01  AE  0365   130$:  MNEGW   #1,NODE$W_BAL(R1)       ; MARK LEFT SIDE HEAVY
        60    11  0369          BRB     240$                    ; SET NEW SUBTREE AND EXIT
                  036B   ;
                  036B   ;      INSERT NEW NODE INTO RIGHT SUBTREE
                  036B   ;
        59    DD  036B   200$:  PUSHL   R9                      ; SAVE POINTER OF FATHER
  59   04 A9  D0  036D          MOVL    NODE$L_RIGHT(R9),R9     ; SETUP POINTER TO RIGHT SON
       FF6F   30  0371          BSBW    INSERT                  ; INSERT INTO LEFT SUBTREE
    51    59  D0  0374          MOVL    R9,R1                   ; SAVE NEW ADDRESS OF SUBTREE
        59  8ED0  0377          POPL    R9                      ; RETURN TO FATHER NODE
  04 A9   51  D0  037A          MOVL    R1,NODE$L_RIGHT(R9)     ; POINT TO NEW RIGHT SUBTREE
     52   50  E9  037E          BLBC    R0,90$                  ; EXIT IF BALANCED
                  0381   ;
                  0381   ;      THE RIGHT SUBTREE HAS GROWN HIGHER - RESTORE BALANCE
                  0381   ;
     08 A9    B6  0381          INCW    NODE$W_BAL(R9)          ; PERFORM RIGHT SHIFT OF TREE
        4B    13  0384          BEQL    80$                     ; EXIT IF IN PERFECT BALANCE
  49 08 A9    E8  0386          BLBS    NODE$W_BAL(R9),90$      ; EXIT IF AVL BLANACED
  51   04 A9  D0  038A          MOVL    NODE$L_RIGHT(R9),R1     ; R1 = POINTER TO RIGHT SUBTREE
     08 A1    B5  038E          TSTW    NODE$W_BAL(R1)          ; TEST BALANCE ON THAT SIDE
        0F    15  0391          BLEQ    220$                    ; BRANCH IF NEED DOUBLE ROTATION
                  0393   ;      PERFORM SINGLE RR ROTATION
  04 A9   61  D0  0393          MOVL    NODE$L_LEFT(R1),NODE$L_RIGHT(R9)
    61    59  D0  0397          MOVL    R9,NODE$L_LEFT(R1)
     08 A9    B4  039A          CLRW    NODE$W_BAL(R9)          ; MARK IN BALANCE
    59    51  D0  039D          MOVL    R1,R9                   ; SET NEW ROOT OF SUBTREE
        2C    11  03A0          BRB     250$                    ; INDICATE IN PERFECT BALANCE
                  03A2   ;      PERFORM DOUBLE RL ROTATION
    52    61  D0  03A2   220$:  MOVL    NODE$L_LEFT(R1),R2      ; R2 = LEFT SON OF RIGHT SUBTREE
  61   04 A2  D0  03A5          MOVL    NODE$L_RIGHT(R2),NODE$L_LEFT(R1)
  04 A2   51  D0  03A9          MOVL    R1,NODE$L_RIGHT(R2)
  04 A9   62  D0  03AD          MOVL    NODE$L_LEFT(R2),NODE$L_RIGHT(R9)
    62    59  D0  03B1          MOVL    R9,NODE$L_LEFT(R2)
     08 A9    B4  03B4          CLRW    NODE$W_BAL(R9)          ; INITIALIZE BALANCE INDICATORS
     08 A1    B4  03B7          CLRW    NODE$W_BAL(R1)
     08 A2    B5  03BA          TSTW    NODE$W_BAL(R2)          ; CHECK BALANCE OF NEW SUBTREE
        0C    13  03BD          BEQL    240$                    ; SKIP IF IN BALANCE
```

```
              06    14  03BF   822         BGTR    230$                    ; BRANCH IF LEFT SIDE HEAVY
       08 A1  01    B0  03C1   823         MOVW    #1,NODE$W_BAL(R1)       ; MARK RIGHT SIDE HEAVY
              04    11  03C5   824         BRB     240$                    ; SET NEW SUBTREE AND EXIT
       08 A9  01    AE  03C7   825 230$:   MNEGW   #1,NODE$W_BAL(R9)       ; MARK LEFT SIDE HEAVY
                        03CB   826 ;
                        03CB   827 ;       SET NEW POINTER TO SUBTREE
                        03CB   828 ;
           59 52    D0  03CB   829 240$:   MOVL    R2,R9                   ; SET NEW SUBTREE
                        03CE   830 ;
                        03CE   831 ;       MARK SUBTREE IN PERFECT BALANCE
                        03CE   832 ;
           08 A9    B4  03CE   833 250$:   CLRW    NODE$W_BAL(R9)          ; MARK IN PERFECT BALANCE
                        03D1   834 ;
                        03D1   835 ;       RETURN R0 FALSE TO INDICATE NEITHER SIDE IS HEAVY
                        03D1   836 ;
              50    D4  03D1   837 80$:    CLRL    R0                      ; MARK IN BALANCE
                    05  03D3   838 90$:    RSB
```

```
                            03D4         .SBTTL  ALLOCATE, ALLOCATE DYNAMIC MEMORY
                            03D4   ;---
                            03D4   ;
                            03D4   ;       THIS ROUTINE ALLOCATES DYNAMIC MEMORY BY EXPANDING THE
                            03D4   ;       REGION IF NECESSARY TO OBTAIN THE REQUESTED AMOUNT OF
                            03D4   ;       MEMORY.
                            03D4   ;
                            03D4   ; INPUTS:
                            03D4   ;
                            03D4   ;       4(AP) = SIZE OF BLOCK TO ALLOCATE
                            03D4   ;
                            03D4   ; OUTPUTS:
                            03D4   ;
                            03D4   ;       R0 = STATUS
                            03D4   ;       R1 = ADDRESS OF ALLOCATED BLOCK
                            03D4   ;
                            03D4   ;       ANY ERRORS ARE SIGNALLED IMMEDIATELY.
                            03D4   ;---
                            03D4
                     0000   03D4         .ENTRY  ALLOCATE,0
                            03D6
               7E      DF   03D6         PUSHAL  -(SP)                     ; RECEIVES ADDRESS OF MEMORY
            04 AC      DF   03D8         PUSHAL  4(AP)                     ; LENGTH TO ALLOCATE
  00000000'GF  02      FB   03DB         CALLS   #2,G^LIB$GET_VM           ; ALLOCATE MEMORY
                            03E2         SIGNAL                            ; SIGNAL IF NO STORAGE
          51   8E      D0   03EE         MOVL    (SP)+,R1                  ; RETURN ADDRESS TO CALLER
                       04   03F1         RET                               ; RETURN WITH STATUS
```

```
                            03F2          .SBTTL  DEALLOCATE, DEALLOCATE DYNAMIC MEMORY
                            03F2   ;---
                            03F2   ;
                            03F2   ;       DEALLOCATE DYNAMIC MEMORY
                            03F2   ;
                            03F2   ; INPUTS:
                            03F2   ;
                            03F2   ;       4(AP) = STARTING ADDRESS TO DEALLOCATE
                            03F2   ;       8(AP) = LENGTH TO DEALLOCATE
                            03F2   ;
                            03F2   ; OUTPUTS:
                            03F2   ;
                            03F2   ;       R0 = STATUS
                            03F2   ;
                            03F2   ;       ANY ERROR IS SIGNALLED IMMEDIATELY.
                            03F2   ;---
                            03F2
                    0000    03F2          .ENTRY  DEALLOCATE,0
                            03F4
               04 AC   DF   03F4          PUSHAL  4(AP)                   ; STARTING ADDRESS
               08 AC   DF   03F7          PUSHAL  8(AP)                   ; LENGTH
    00000000'GF   02   FB   03FA          CALLS   #2,G^LIB$FREE_VM        ; DEALLOCATE STORAGE
                            0401          SIGNAL                          ; SIGNAL ANY ERRORS
                      04    040D          RET
```

```
                      040E        .SBTTL  READ_SYMBOLS -- READ STB SYMBOLS INTO SYMBOL TABLE
                      040E  ;---
                      040E  ;
                      040E  ;     READ_SYMBOLS
                      040E  ;
                      040E  ;     THIS ROUTINE READS ALL THE SYSTEM SYMBOLS INTO DYNAMIC
                      040E  ;     STORAGE AND CREATES A LIST IN ASCENDING ORDER BY
                      040E  ;     SYMBOL VALUE.  THIS MAKES IT POSSIBLE TO TRANSFORM
                      040E  ;     A GIVEN VALUE INTO A SYMBOL NAME AND AN OFFSET BY
                      040E  ;     SCANNING THE SYMBOL TABLE UNTIL THE CORRECT AREA
                      040E  ;     IS DETERMINED.
                      040E  ;
                      040E  ; INPUTS:
                      040E  ;
                      040E  ;     NONE
                      040E  ;
                      040E  ; OUTPUTS:
                      040E  ;
                      040E  ;     SYMFL = LIST HEADER FOR SORTED VALUE LIST
                      040E  ;     ALPFL = LIST HEADER FOR SORTED ALPHABETIC LIST
                      040E  ;
                      040E  ;---
                      040E
                 0FFC 040E        .ENTRY  READ_SYMBOLS,-
                      0410                ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                      0410
      FD23 CF   00  FB 0410        CALLS   #0,REWIND_STB            ; REWIND THE STB FILE
      00000014'EF   D4 0415        CLRL    NSYMBOLS                 ; ZERO NUMBER OF SYMBOLS
      0000000C'EF   D4 041B        CLRL    SYMFL                    ; INITIALIZE TREE ROOTS
      00000010'EF   D4 0421        CLRL    ALPFL
                      0427  ;
                      0427  ;     READ GLOBAL SYMBOLS FROM SYS.STB INTO SYMBOL TABLE
                      0427  ;
                      0427  10$:
      FD15 CF   00  FB 0427        CALLS   #0,GET_SYMBOL            ; GET NEXT SYMBOL ENTRY
           10   50  E9 042C        BLBC    R0,20$                   ; BRANCH IF END OF FILE
   53      09   A1  9E 042F        MOVAB   OBJ$T_SYM_NAME(R1),R3    ; GET ADDRESS OF ASCIC SYMBOL NAME
        52      83  9A 0433        MOVZBL  (R3)+,R2                 ; GET LENGTH AND ADDRESS
   54      05   A1  D0 0436        MOVL    OBJ$L_SYM_VALUE(R1),R4   ; GET VALUE OF SYMBOL
              FE3B  30 043A        BSBW    ADD_NEW_ENTRY            ; ADD NEW SYMBOL (ASSUME NOT IN TABLE)
                E8  11 043D        BRB     10$                      ; CONTINUE UNTIL ALL SYMBOLS READ
                      043F  20$:
                      043F        $CLOSE  STBF                     ; CLOSE SYSTEM SYMBOLS FILE
                      044C  ;
                      044C  ;     SAVE SPECIFIC SYMBOL VALUES IN REQUIRED SYMBOLS TABLE
                      044C  ;
   56   00000000'EF     9E 044C        MOVAB   REQ_SYMBOLS,R6           ; INITIALIZE TABLE POINTER
   57      067C C6      9E 0453        MOVAB   REQ_SYMBOLS_LEN(R6),R7   ; ENDING ADDRESS OF TABLE
           05   A6  9F 0458  30$:  PUSHAB  5(R6)                    ; CONSTRUCT STRING DESCRIPTORR
   7E      04   A6  9A 045B        MOVZBL  4(R6),-(SP)
                5E  DD 045F        PUSHL   SP                       ; ADDRESS OF DESCRIPTOR
 000007BA'EF    03  FB 0461        CALLS   #3,SYMBOL_VALUE          ; LOOKUP SYMBOL
           05   50  E9 0468        BLBC    R0,40$                   ; BRANCH IF NOT FOUND
        66      51  D0 046B        MOVL    R1,(R6)                  ; STORE VALUE OF SYMBOL
                18  11 046E        BRB     50$                      ; GO ON TO NEXT SYMBOL
                      0470  40$:
           15   66  E8 0470        BLBS    (R6),50$                 ; IGNORE IF OPTIONAL SYMBOL
```

```
             04 A6   9F  0473              PUSHAB  4(R6)                   ; ADDRESS OF COUNTED STRING
                         0476              SIGNAL  1,NOREQ                 ; SIGNAL SYMBOL NOT FOUND
                         0488      50$:
       50    04 A6   9A  0488              MOVZBL  4(R6),R0                ; LENGTH OF SYMBOL
     56    05 A640   9E  048C              MOVAB   5(R6)[R0],R6            ; INCREMENT POINTER
          57     56  D1  0491              CMPL    R6,R7                   ; CHECK IF END OF TABLE
                 C2  1F  0494              BLSSU   30$                     ; CONTINUE IF MORE TO DO
                         0496      ;
                         0496      ;       DEFINE SDA PERMANENT SYMBOLS (R0, ETC.)
                         0496      ;
     52    FB66 CF   9E  0496              MOVAB   REG_TABLE,R2            ; ADDRESS OF REGISTER TABLE
                         049B      70$:
          53     82  7D  049B              MOVQ    (R2)+,R3                ; GET SYMBOL ADDRESS, VALUE
                 0C  13  049E              BEQL    80$                     ; BRANCH IF END OF TABLE
                 54  DD  04A0              PUSHL   R4                      ; VALUE OF SYMBOL
          7E     63  7D  04A2              MOVQ    (R3),-(SP)              ; DESCRIPTOR OF SYMBOL
     FD6B CF     03  FB  04A5              CALLS   #3,ADD_SYMBOL           ; ADD TO SYMBOL TABLE
                 EF  11  04AA              BRB     70$                     ; CONTINUE UNTIL DONE
                         04AC      ;
                         04AC      ;       DEFINE SPECIAL SYMBOL "RMS" TO BE BASE ADDRESS OF RMS
                         04AC      ;
                         04AC      80$:    REQMEM  @MMG$GL_RMSBASE,-(SP)   ; GET BASE OF RMS CODE
  7E    0000067C'EF  7D  04BC              MOVQ    RMS,-(SP)               ; DESCRIPTOR OF RMS SYMBOL
     FD4D CF     03  FB  04C3              CALLS   #3,ADD_SYMBOL           ; ADD TO SYMBOL TABLE
                         04C8      ;
                         04C8      ;       DEFINE SPECIAL SYMBOL "MP" TO BE BASE ADDRESS OF MULTI PROCESSOR
                         04C8      ;       CODE.
                         04C8      ;
                         04C8              REQMEM  @EXE$GL_MP,-(SP)        ; GET BASE OF MP CODE
  7E    00000693'EF  7D  04D8              MOVQ    MP,-(SP)                ; DESCRIPTOR OF MP SYMBOL
     FD31 CF     03  FB  04DF              CALLS   #3,ADD_SYMBOL           ; ADD TO SYMBOL TABLE
                         04E4      ;
                         04E4      ;       DEFINE SPECIAL SYMBOL "MSCP" TO BE BASE ADDRESS OF MSCP SERVER
                         04E4      ;       CODE.
                         04E4      ;
                         04E4              REQMEM  @SCS$GL_MSCP,-(SP)      ; GET BASE OF MSCP CODE
  7E    000006AE'EF  7D  04F4              MOVQ    MSCP,-(SP)              ; DESCRIPTOR OF MSCP SYMBOL
     FD15 CF     03  FB  04FB              CALLS   #3,ADD_SYMBOL           ; ADD TO SYMBOL TABLE
                         0500      ;
                         0500      ;       DEFINE SPECIAL SYMBOL "CLUSTRLOA" TO BE BASE ADDRESS OF
                         0500      ;       CLUSTRLOA CODE.
                         0500      ;
                         0500              REQMEM  @CLU$GL_LOA_ADDR,-(SP)  ; GET BASE OF CLUSTRLOA CODE
  7E    0000069D'EF  7D  0510              MOVQ    CLUSTRLOA,-(SP)         ; DESCRIPTOR OF CLUSTRLOA SYMBOL
     FCF9 CF     03  FB  0517              CALLS   #3,ADD_SYMBOL           ; ADD TO SYMBOL TABLE
                         051C      ;
                         051C      ;       DEFINE SPECIAL SYMBOL "SYSLOA" TO BE BASE ADDRESS OF
                         051C      ;       SYSLOA CODE.
                         051C      ;
                         051C      ;       REQMEM  @MMG$GL_SYSLOA_BASE,-(SP)       ; GET BASE OF SYSLOA CODE
                         051C      ;       MOVQ    SYSLOA,-(SP)            ; DESCRIPTOR OF SYSLOA SYMBOL
                         051C      ;       CALLS   #3,ADD_SYMBOL           ; ADD TO SYMBOL TABLE
                         051C      ;
                         051C      ;       DEFINE SPECIAL SYMBOL "SCSLOA" TO BE BASE ADDRESS OF
                         051C      ;       SCSLOA CODE.
                         051C      ;
                         051C              REQMEM  @SCS$GA_EXISTS,-(SP)    ; GET BASE OF SCSLOA CODE
```

```
7E   000006E5'EF  7D 052C        MOVQ    SCSLOA,-(SP)                   ; DESCRIPTOR OF SCSLOA SYMBOL
     FCDD CF      03 FB 0533     CALLS   #3,ADD_SYMBOL                  ; ADD TO SYMBOL TABLE
                     0538  ;
                     0538  ;     DEFINE SPECIAL SYMBOL "VAXEMUL" TO BE BASE ADDRESS OF
                     0538  ;     VAX EMULATOR CODE.
                     0538  ;
                     0538        REQMEM  @MMG$GL_VAXEMUL_BASE,-(SP)          ; GET BASE OF VAXEMUL CODE
                     0538        MOVQ    VAXEMUL,-(SP)                  ; DESCRIPTOR OF VAXEMUL SYMBOL
                     0538        CALLS   #3,ADD_SYMBOL                  ; ADD TO SYMBOL TABLE
                     0538  ;
                     0538  ;     DEFINE SPECIAL SYMBOL "FPEMUL" TO BE BASE ADDRESS OF
                     0538  ;     FLOATING POINT EMULATOR CODE.
                     0538  ;
                     0538        REQMEM  @MMG$GL_FPEMUL_BASE,-(SP)           ; GET BASE OF FPEMUL CODE
                     0538        MOVQ    FPEMUL,-(SP)                   ; DESCRIPTOR OF FPEMUL SYMBOL
                     0538        CALLS   #3,ADD_SYMBOL                  ; ADD TO SYMBOL TABLE
                     0538
                     0538  ;
                     0538  ;     ADD SPECIAL SYMBOLS MARKING THE STARTING ADDRESS OF EACH DRIVER.
                     0538  ;     EACH SYMBOL WILL MARK THE START OF THE DRIVER PROLOGUE TABLE (DPT).
                     0538  ;
        5E 38  C2    0538  90$:  SUBL    #DPT$C_LENGTH,SP               ; ALLOCATE BUFFER TO READ DPT'S
        57 5E  D0    053B        MOVL    SP,R7                          ; AND SAVE ADDRESS OF IT
                     053E        TRYMEM  @IOC$GL_DPTLIST,DPT$L_FLINK(R7) ; SET ADDRESS OF FIRST DPT
        2C 50  E9    054E        BLBC    R0,100$                        ; BRANCH IF ERROR
        58 67  D0    0551  95$:  MOVL    DPT$L_FLINK(R7),R8             ; SKIP TO NEXT DPT
00000142'EF 58 D1    0554        CMPL    R8,IOC$GL_DPTLIST              ; CHECK IF BACK TO LISTHEAD
           20  13    055B        BEQL    100$                           ; BRANCH IF END OF LIST
                     055D        TRYMEM  (R8),(R7),#DPT$C_LENGTH        ; READ THE ENTIRE DPT
        10 50  E9    056A        BLBC    R0,100$                        ; BRANCH IF ERROR
           58  DD    056D        PUSHL   R8                             ; PUSH REAL ADDRESS OF ACTUAL DPT
        21 A7  9F    056F        PUSHAB  DPT$T_NAME+1(R7)               ; PUSH DESCRIPTOR OF DRIVER NAME
   7E   20 A7  9A    0572        MOVZBL  DPT$T_NAME(R7),-(SP)
     FC9A CF   03 FB 0576        CALLS   #3,ADD_SYMBOL                  ; ADD TO SYMBOL TABLE
           D4  11    057B        BRB     95$
                     057D  ;
                     057D  ; ADD SPECIAL SYMBOLS FOR CPU-SPECIFIC LOADABLE CODE
                     057D  ;
50   00000122'EF  D0 057D  100$: MOVL    EXE$MCHK,R0                    ; GET ADDRESS OF EXE$MCHK VECTOR
                     0584                                               ; (OF THE FORM: JMP @#MCHK_CODE)
                     0584        REQMEM  2(R0),-(SP)                    ; GET ACTUAL ADDRESS OF MCHK CODE
7E   00000687'EF  7D 0591        MOVQ    MCHK,-(SP)                     ; PUSH DESCRIPTOR OF MCHK SYMBOL
     FC78 CF      03 FB 0598     CALLS   #3,ADD_SYMBOL                  ; ADD TO SYMBOL TABLE
                     059D
                     059D        STATUS  SUCCESS
                  04 05A4        RET
```

```
                             05A5            .SBTTL  PRINT_SYMBOLS -- PRINT ALL SYSTEM SYMBOLS
                             05A5       ;---
                             05A5       ;
                             05A5       ;       PRINT_SYMBOLS
                             05A5       ;
                             05A5       ;       THIS ROUTINE PRINTS ALL THE SYSTEM SYMBOLS IN ASCENDING
                             05A5       ;       VALUE ORDER AND THEIR ASSOCIATED VALUE.  IF A GENERIC
                             05A5       ;       NAME IS SUPPLIED, THEN PRINT ONLY THOSE SYMBOLS STARTING
                             05A5       ;       WITH THE GIVEN STRING.
                             05A5       ;
                             05A5       ;   INPUTS:
                             05A5       ;
                             05A5       ;       AP = ADDRESS OF DESCRIPTOR OF GENERIC NAME, IF ANY.
                             05A5       ;       ALPFL = LIST HEADER FOR SORTED ALPHABETIC LIST
                             05A5       ;       SYMFL = LIST HEADER FOR SORTED VALUE LIST
                             05A5       ;
                             05A5       ;   OUTPUTS:
                             05A5       ;
                             05A5       ;       NONE
                             05A5       ;
                             05A5       ;---
                             05A5
            00000024         05A5       PRTBUFLEN       = 36                    ; CHARACTERS PER LINE SEGMENT
            0000002C         05A5       PRTBUFSIZ       = 8+PRTBUFLEN           ; BYTES PER SEGMENT BUFFER
            00000036         05A5       ROWS            = 54                    ; ROWS PER COLUMN
            00000003         05A5       COLUMNS         = 3                     ; COLUMNS PER PAGE
            00001BD8         05A5       STORAGE_SIZE    = ROWS*COLUMNS*PRTBUFSIZ ; # BYTES FOR ONE PAGE
                             05A5
                             05A5       PRINT_SYMBOLS::
                   0FDC      05A5               .WORD   ^M<R2,R3,R4,R6,R7,R8,R9,R10,R11>
                             05A7
                             05A7               SUBHD   <Symbols sorted by name>
                             05B4               SKIP    PAGE
     59   00000010'EF   D0   05BB               MOVL    ALPFL,R9                ; ROOT OF VALUE TREE
                   1E   10   05C2               BSBB    PRINT_LIST              ; PRINT ENTIRE LIST
                             05C4
                             05C4               SUBHD   <Symbols sorted by value>
                             05D1               SKIP    PAGE
     59   0000000C'EF   D0   05D8               MOVL    SYMFL,R9                ; ROOT OF ALPHABETIC TREE
                   01   10   05DF               BSBB    PRINT_LIST              ; PRINT ENTIRE LIST
                        04   05E1               RET
                             05E2
                             05E2       ;
                             05E2       ;       PRINT ENTIRE SYMBOL LIST
                             05E2       ;
                             05E2       ;       AP = ADDRESS OF DESCRIPTOR OF GENERIC NAME
                             05E2       ;       R2 = ADDRESS OF LIST HEADER
                             05E2       ;       R3 = LONGWORD INDEX TO LINK WITHIN SYMBOL RECORD
                             05E2       ;
                             05E2
                             05E2       PRINT_LIST:
      00001BD8 8F       DD   05E2               PUSHL   #STORAGE_SIZE           ; LENGTH OF BUFFER STORAGE
      FDE7 CF   01      FB   05E8               CALLS   #1,ALLOCATE             ; ALLOCATE BUFFER STORAGE
            57   51     D0   05ED               MOVL    R1,R7                   ; SAVE ADDRESS OF BUFFER MATRIX
     50    00A2 8F      3C   05F0               MOVZWL  #ROWS*COLUMNS,R0        ; REPEAT COUNT
            54   57     D0   05F5               MOVL    R7,R4                   ; ADDRESS OF BUFFER MATRIX
                             05F8       5$:
```

```
             64   24   D0  05F8          MOVL    #PRTBUFLEN,(R4)              ; INITIALIZE DESCRIPTOR
      04 A4   08 A4   DE  05FB          MOVAL   8(R4),4(R4)
             54   2C   C0  0600          ADDL2   #PRTBUFSIZ,R4
             F2   50   F5  0603          SOBGTR  R0,5$                        ; INIT. ALL DESCRIPTORS
       000006C7'EF     9F  0606          PUSHAB  TRAVERSE                     ; INIT CO-ROUTINE ADDRESS
                           060C
                           060C  10$:
             54   57   D0  060C          MOVL    R7,R4                        ; ADDRESS OF BUFFER MATRIX
      56   00A2 8F     3C  060F          MOVZWL  #ROWS*COLUMNS,R6             ; LOOP COUNT
                           0614  20$:
                  9E   16  0614          JSB     @(SP)+                       ; GET THE NEXT ENTRY
             65   50   E9  0616          BLBC    R0,90$                       ; BRANCH IF DONE
         52   0A A9   D0  0619          MOVL    NODE$L_PTR(R9),R2            ; ADDRESS OF SYMBOL TABLE ENTRY
     C0000000 8F  62   D1  061D          CMPL    SYM$L_VALUE(R2),#^XC0000000  ; CHECK IF INTERNAL SYMBOL
                  EE   1E  0624          BGEQU   20$                          ; SKIP SYMBOL IF SO
                  6C   B5  0626          TSTW    (AP)                         ; ANY GENERIC NAME?
                  10   13  0628          BEQL    25$                          ; BRANCH IF NOT
                  3C   BB  062A          PUSHR   #^M<R2,R3,R4,R5>             ; SAVE REGISTERS FOR CMPC
   05 A2    04 BC    6C   29  062C          CMPC    (AP),@4(AP),SYM$T_SYMBOL+1(R2) ; GENERIC MATCH?
                  04   13  0632          BEQL    22$                          ; BRANCH IF MATCH
                  3C   BA  0634          POPR    #^M<R2,R3,R4,R5>             ; RESTORE REGISTERS
                  DC   11  0636          BRB     20$                          ; KEEP LOOKING FOR MATCH
                  3C   BA  0638  22$:    POPR    #^M<R2,R3,R4,R5>             ; RESTORE REGISTERS
                           063A  25$:    TRYMEM  @SYM$L_VALUE(R2)             ; ATTEMPT TO READ CONTENTS
             18   50   E9  0644          BLBC    R0,30$                       ; IF LOCATION CANNOT BE READ
                  51   DD  0647          PUSHL   R1                           ; CONTENTS OF LOCATION
                  62   DD  0649          PUSHL   SYM$L_VALUE(R2)              ; SYMBOL VALUE
             04   A2   9F  064B          PUSHAB  SYM$T_SYMBOL(R2)             ; SYMBOL NAME
                  54   DD  064E          PUSHL   R4                           ; BUFFER DESCRIPTOR
                  00   DD  0650          PUSHL   #0
           FAA5   CF   9F  0652          PUSHAB  PRTCTL2
       00000000'GF     06   FB  0656          CALLS   #6,G^SYS$FAO                 ; FORMAT STRING
                  14   11  065D          BRB     50$
                           065F  30$:
                  62   DD  065F          PUSHL   SYM$L_VALUE(R2)              ; SYMBOL VALUE
             04   A2   9F  0661          PUSHAB  SYM$T_SYMBOL(R2)             ; SYMBOL NAME
                  54   DD  0664          PUSHL   R4                           ; BUFFER DESCRIPTOR
                  00   DD  0666          PUSHL   #0
           FA7C   CF   9F  0668          PUSHAB  PRTCTL1                      ; CONTROL STRING
       00000000'GF     05   FB  066C          CALLS   #5,G^SYS$FAO                 ; FORMAT STRING
                           0673  50$:
             54   2C   C0  0673          ADDL2   #PRTBUFSIZ,R4                ; NEXT OUTPUT DESCRIPTOR
             9B   56   F5  0676          SOBGTR  R6,20$                       ; CONTINUE UNTIL PAGE FULL
                  1B   10  0679          BSBB    PRINT_PAGE                   ; PRINT ENTIRE PAGE
                FF8E   31  067B          BRW     10$                          ; CONTINUE UNTIL DONE
                           067E  90$:
                  64   D4  067E          CLRL    (R4)                         ; ZERO DESCRIPTOR LENGTH
             54   2C   C0  0680          ADDL2   #PRTBUFSIZ,R4
             F8   56   F5  0683          SOBGTR  R6,90$                       ; ZERO REMAINING DESCRIPTORS
                  0E   10  0686          BSBB    PRINT_PAGE                   ; PRINT LAST PAGE
        00001BD8 8F    DD  0688          PUSHL   #STORAGE_SIZE                ; LENGTH TO DEALLOCATE
                  57   DD  068E          PUSHL   R7                           ; ADDRESS OF BUFFER STORAGE
        FD5D CF   02   FB  0690          CALLS   #2,DEALLOCATE                ; RELEASE STORAGE SPACE
                       05  0695          RSB
                           0696
                           0696  ;
                           0696  ;       PRINT ENTIRE PAGE OF DESCRIPTORS
```

```
                          0696  ;
                          0696
                          0696          .ENABL  LSB
                          0696
                          0696  PRINT_PAGE:
            56   36   D0  0696          MOVL    #ROWS,R6                        ; LINES TO PRINT
            54   57   D0  0699          MOVL    R7,R4                           ; R4 => 1ST SEGMENT IN EACH ROW
                          069C  10$:
  50   0948 C4   64   A1  069C          ADDW3   (R4),ROWS*PRTBUFSIZ(R4),R0 ; ADD FIRST 2 LENGTHS
     50   1290 C4   A0    06A2          ADDW    2*ROWS*PRTBUFSIZ(R4),R0     ; ADD IN THIRD LENGTH
                 17   13  06A7          BEQL    20$                         ; SKIP NULL LINES
          1290 C4   7F    06A9          PUSHAQ  <2*ROWS*PRTBUFSIZ>(R4)
          0948 C4   7F    06AD          PUSHAQ  <1*ROWS*PRTBUFSIZ>(R4)
                 64   7F  06B1          PUSHAQ  (R4)
                          06B3          PRINT   3,<!AS!8* !AS!8* !AS>       ; PRINT LINE
            54   2C   C0  06C0  20$:    ADDL2   #PRTBUFSIZ,R4               ; SKIP TO NEXT ROW
               D6   56  F5  06C3        SOBGTR  R6,10$                      ; LOOP UNTIL DONE
                      05  06C6          RSB
                          06C7
                          06C7          .DSABL  LSB
```

```
                    06C7          .SBTTL  TRAVERSE, COROUTINE TO TRAVERSE A TREE
                    06C7   ;---
                    06C7   ;
                    06C7   ;       TRAVERSE
                    06C7   ;
                    06C7   ;       THIS COROUTINE RETURNS THE NEXT ENTRY IN A TREE BY
                    06C7   ;       TRAVERSING THE TREE IN PREORDER FASHION.  THE COROUTINE
                    06C7   ;       TERMINATES AND RETURNS WITH R0 FALSE IF NO MORE ENTRIES
                    06C7   ;       EXIST.
                    06C7   ;
                    06C7   ; INPUTS:
                    06C7   ;
                    06C7   ;       R9  = ADDRESS OF CURRENT NODE
                    06C7   ;
                    06C7   ; OUTPUTS:
                    06C7   ;
                    06C7   ;       R9 = ADDRESS OF NEXT NODE IN SEQUENCE
                    06C7   ;       R0 = TRUE IF MORE TO GO, FALSE IF DONE
                    06C7   ;
                    06C7   ;       R11 IS DESTROYED.
                    06C7   ;
                    06C7   ;---
                    06C7
                    06C7   TRAVERSE:
       5B   6E  D0  06C7          MOVL    (SP),R11                ; SAVE RETURN ADDRESS
            59  DD  06CA   10$:   PUSHL   R9                      ; SAVE FATHER
            18  13  06CC          BEQL    90$                     ; EXIT IF NULL NODE
       59   69  D0  06CE          MOVL    NODE$L_LEFT(R9),R9      ; POINT TO LEFT SUBTREE
            F7  10  06D1          BSBB    10$                     ; TRAVERSE THE LEFT HALF
          59  8ED0  06D3          POPL    R9                      ; RESTORE FATHER
       50   01  D0  06D6          MOVL    #1,R0                   ; TELL USER MORE TO COME
            6B  16  06D9          JSB     (R11)                   ; CALL THE USER WITH NODE
       5E   04  C0  06DB          ADDL    #4,SP                   ; THROW THE RETURN ADDRESS AWAY
            59  DD  06DE          PUSHL   R9                      ; SAVE FATHER
  59   04   A9  D0  06E0          MOVL    NODE$L_RIGHT(R9),R9     ; POINT TO RIGHT SUBTREE
            E4  10  06E4          BSBB    10$                     ; TRAVERSE THE RIGHT HALF
          59  8ED0  06E6   90$:   POPL    R9                      ; RESTORE FATHER
            50  D4  06E9          CLRL    R0                      ; INDICATE NO MORE
                05  06EB          RSB
```

```
                          06EC          .SBTTL  ALP_TRAVERSE -- TRAVERSE ALPHA TREE ON PREFIX MATCH
                          06EC  ;---
                          06EC  ;
                          06EC  ;       ALP_TRAVERSE
                          06EC  ;
                          06EC  ;       This is a coroutine, like TRAVERSE, which scans the tree in
                          06EC  ;       postorder, making a cocall on each node until the end of the
                          06EC  ;       tree is reached.  It takes as an argument a prefix string
                          06EC  ;       and only returns those nodes which match the prefix string.
                          06EC  ;
                          06EC  ;       INPUTS:
                          06EC  ;
                          06EC  ;       R6      Length of prefix string
                          06EC  ;       R7      Address of each prefix string
                          06EC  ;       R9      Listhead node
                          06EC  ;
                          06EC  ;       OUTPUTS:
                          06EC  ;
                          06EC  ;       R9      Address of current node
                          06EC  ;       R10     Symbol block address for this node
                          06EC  ;       R0      TRUE iff R9 contains a valid node address and scan hasn't ended
                          06EC  ;
                          06EC  ;       R11     is destroyed
                          06EC  ;---
                          06EC
                          06EC  ALP_TRAVERSE:
          5B    6E    D0  06EC          MOVL    (SP),R11                        ; Save return address
                          06EF  10$:
                59    D5  06EF          TSTL    R9                              ; Reached terminal node?
                39    13  06F1          BEQL    40$                             ; Return if so
          5A  0A A9   D0  06F3          MOVL    NODE$L_PTR(R9),R10              ; Get symbol block
 05 AA    67    56    29  06F7          CMPC3   R6,(R7),SYM$T_SYMBOL+1(R10)     ; Compare prefix string
                0D    13  06FC          BEQL    30$                             ; J if exact match, must return node
                06    1F  06FE          BLSSU   20$                             ; J if less
          59  04 A9   D0  0700          MOVL    NODE$L_RIGHT(R9),R9             ; Greater - go right
                E9    11  0704          BRB     10$                             ; and loop
                          0706  20$:
          59    69    D0  0706          MOVL    NODE$L_LEFT(R9),R9              ; Less - go left
                E4    11  0709          BRB     10$                             ; and loop
                          070B  30$:
                59    DD  070B          PUSHL   R9                              ; Equal - save this node
          59    69    D0  070D          MOVL    NODE$L_LEFT(R9),R9              ; Scan left subtree
                DD    10  0710          BSBB    10$                             ; ...
               59   8ED0  0712          POPL    R9                              ; retrieve current node
          5A  0A A9   D0  0715          MOVL    NODE$L_PTR(R9),R10              ; Get symbol block
          50    01    D0  0719          MOVL    #1,R0                           ; Show node present
                6B    16  071C          JSB     (R11)                           ; Cocall our caller
          5B    8E    D0  071E          MOVL    (SP)+,R11                       ; Replace return address
                59    DD  0721          PUSHL   R9                              ; Save R9 again
          59  04 A9   D0  0723          MOVL    NODE$L_RIGHT(R9),R9             ; Now scan right subtree
                C6    10  0727          BSBB    10$                             ; ...
               59   8ED0  0729          POPL    R9                              ; Recover node address
                50    D4  072C  40$:    CLRL    R0                              ; Show no more to scan
                      05  072E          RSB                                     ; And get out
```

```
                              072F          .SBTTL  SYMBOLIZE -- CONVERT VALUE TO SYMBOL AND OFFSET
                              072F     ;---
                              072F     ;
                              072F     ;        SYMBOLIZE
                              072F     ;
                              072F     ;        THIS ROUTINE ACCEPTS A VALUE AND RETURNS A STRING WHICH
                              072F     ;        CONTAINS A SYMBOL AND OFFSET CORRESPONDING TO THAT VALUE.
                              072F     ;
                              072F     ;   INPUTS:
                              072F     ;
                              072F     ;        4(AP)  = VALUE TO BE TRANSLATED
                              072F     ;        8(AP)  = ADDRESS OF QUADWORD RESULT STRING DESCRIPTOR
                              072F     ;
                              072F     ;   OUTPUTS:
                              072F     ;
                              072F     ;        THE STRING IS RETURNED IN THE RESULT BUFFER
                              072F     ;        R1     = ADDRESS OF DESCRIPTOR OF RESULT STRING
                              072F     ;
                              072F     ;---
                              072F
                              072F     SYMBOLIZE::
                        0A0C  072F              .WORD   ^M<R2,R3,R9,R11>
                              0731
     53   00000004'EF    DE   0731              MOVAL   TRANSLTH,R3
             63   08 BC  7D   0738              MOVQ    @8(AP),(R3)                     ; PRESET RESULT DESCRIPTOR
                              073C
        15 04 AC    1F   E0   073C              BBS     #31,4(AP),20$                   ; BRANCH IF SYSTEM ADDRESS
        10 04 AC    1E   E0   0741              BBS     #30,4(AP),20$                   ; BRANCH IF CONTROL REGION
  00000200 8F    04 AC   D1   0746              CMPL    4(AP),#^X200                    ; IS ADDRESS IN FIRST PAGE?
                    06   1E   074E              BGEQU   20$                             ; IF NOT, ATTEMPT LOOKUP
                              0750     10$:
              51    53   D0   0750              MOVL    R3,R1                           ; R1 -> OUTPUT STRING
                    61   D4   0753              CLRL    (R1)                            ; RETURN NULL STRING
                         04   0755              RET
                              0756     ;
                              0756     ;        Search value ordered tree to find symbol.  The last symbol which
                              0756     ;        we pass on the way down the tree, whose value is less than the
                              0756     ;        value we are looking for, will be the largest symbol in the
                              0756     ;        tree which is less than the value (honest!).
                              0756     ;
                    52   D4   0756     20$:     CLRL    R2                              ; CLEAR 'LAST SMALLER VALUE' POINTER
     59   0000000C'EF    D0   0758              MOVL    SYMFL,R9                        ; GET TREE HEAD
                    1D   13   075F     30$:     BEQL    60$                             ; J IF REACHED TERMINAL NODE
         50    0A A9     D0   0761              MOVL    NODE$L_PTR(R9),R0               ; GET SYMBOL BLOCK
         60    04 AC     D1   0765              CMPL    4(AP),SYM$L_VALUE(R0)           ; COMPARE AGAINST KEY VALUE
                    10   13   0769              BEQL    50$                             ; J IF EXACT MATCH, LOOK NO MORE
                    09   1F   076B              BLSSU   40$                             ; J IS LESS - TAKE LEFT BRANCH
              52    50   D0   076D              MOVL    R0,R2                           ; KEY IS LARGER - REMEMBER SYMBOL
         59    04 A9     D0   0770              MOVL    NODE$L_RIGHT(R9),R9             ; GO RIGHT
                    E9   11   0774              BRB     30$                             ; AND LOOP
              59    69   D0   0776     40$:     MOVL    NODE$L_LEFT(R9),R9              ; KEY IS SMALLER - GO LEFT
                    E4   11   0779              BRB     30$                             ; AND LOOP
              52    50   D0   077B     50$:     MOVL    R0,R2                           ; EXACT MATCH - REMEMBER IT
                              077E
                              077E                                                      ; BEST SYMBOL NOW IN R2
                    52   D5   077E     60$:     TSTL    R2                              ; FOUND A SYMBOL?
                    CE   13   0780              BEQL    10$                             ; STRAIGHT OUT IF NOT
```

```
      51   04 AC    62   C3  0782  1341          SUBL3   SYM$L_VALUE(R2),4(AP),R1 ; OFFSET VALUE
  00000FFF 8F       51   D1  0787  1342          CMPL    R1,#^XFFF             ; MUST NOT BE OVER 3 DIGITS
                    C0   1A  078E  1343          BGTRU   10$                   ; IF SO, DO NOT TRANSLATE
                    51   DD  0790  1344          PUSHL   R1                    ; OFFSET VALUE
              04    A2   9F  0792  1345          PUSHAB  SYM$T_SYMBOL(R2)      ; SYMBOL NAME
              08    AC   DD  0795  1346          PUSHL   8(AP)                 ; RESULT STRING DESCRIPTOR
                    53   DD  0798  1347          PUSHL   R3                    ; RESULT LENGTH
                    51   D5  079A  1348          TSTL    R1                    ; CHECK IF ZERO OFFSET
                    0D   12  079C  1349          BNEQ    70$                   ; BRANCH IF NONZERO
              F981  CF   DF  079E  1350          PUSHAL  TRANSCTL2
  00000000'GF       04   FB  07A2  1351          CALLS   #4,G^SYS$FAO          ; FORMAT OUTPUT STRING
                    0B   11  07A9  1352          BRB     80$
              F964  CF   DF  07AB  1353 70$:     PUSHAL  TRANSCTL1             ; CONTROL STRING
  00000000'GF       05   FB  07AF  1354          CALLS   #5,G^SYS$FAO          ; FORMAT OUTPUT STRING
              51    53   D0  07B6  1355 80$:     MOVL    R3,R1                 ; R1 -> OUTPUT DESCRIPTOR
                         04  07B9  1356          RET
```

```
                          07BA           .SBTTL  SYMBOL_VALUE -- GET VALUE OF SPECIFIED SYMBOL
                          07BA    ;---
                          07BA    ;
                          07BA    ;      SYMBOL_VALUE
                          07BA    ;
                          07BA    ;      THIS ROUTINE RETURNS THE VALUE OF A GIVEN SYMBOL
                          07BA    ;
                          07BA    ;   INPUTS:
                          07BA    ;
                          07BA    ;      4(AP) = ADDRESS OF SYMBOL DESCRIPTOR
                          07BA    ;
                          07BA    ;   OUTPUTS:
                          07BA    ;
                          07BA    ;      R0 = TRUE IF SYMBOL FOUND
                          07BA    ;      R1 = VALUE OF SYMBOL IF FOUND
                          07BA    ;
                          07BA    ;---
                          07BA
                          07BA    SYMBOL_VALUE::
                    03FC  07BA           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                          07BC
            58   04 BC 7D 07BC           MOVQ    @4(AP),R8                     ; GET DESCRIPTOR
      57   00000010'EF D0 07C0           MOVL    ALPFL,R7                      ; ROOT OF TREE
                          07C7    20$:
                    57 D5 07C7           TSTL    R7                            ; NULL POINTER?
                    25 13 07C9           BEQL    80$                           ; IF NOT FOUND
            56   0A A7 D0 07CB           MOVL    NODE$L_PTR(R7),R6             ; ADDRESS OF SYMBOL TABLE ENTRY
            50   04 A6 9A 07CF           MOVZBL  SYM$T_SYMBOL(R6),R0           ; LENGTH OF SYMBOL
 05 A6   50   20   69 58 2D 07D3         CMPC5   R8,(R9),#^A' ',R0,SYM$T_SYMBOL+1(R6)
                    0D 13 07DA           BEQL    30$                           ; BRANCH IF NOT FOUND
                    05 1A 07DC           BGTRU   25$                           ; BRANCH IF GREATER
               57   67 D0 07DE           MOVL    NODE$L_LEFT(R7),R7            ; GO TO LEFT BRANCH OF TREE
                    E4 11 07E1           BRB     20$                           ; CONTINUE DOWN TREE
            57   04 A7 D0 07E3    25$:   MOVL    NODE$L_RIGHT(R7),R7           ; GO TO RIGHT BRANCH
                    DE 11 07E7           BRB     20$                           ; CONTINUE DOWN TREE
               50   01 D0 07E9    30$:   MOVL    #1,R0                         ; SYMBOL FOUND
               51   66 D0 07EC           MOVL    SYM$L_VALUE(R6),R1            ; VALUE OF SYMBOL
                       04 07EF           RET
                          07F0    80$:
                    50 D4 07F0           CLRL    R0                            ; SYMBOL NOT FOUND
                       04 07F2           RET
```

```
                                07F3          .sbttl  FORMAT, Process the FORMAT command
                                07F3    ;---
                                07F3    ;
                                07F3    ;       This routine processes the FORMAT command which
                                07F3    ;       specifies an address and the symbol prefix used
                                07F3    ;       to specify the type of data structure.  The symbol
                                07F3    ;       table will be searched for all symbols containing
                                07F3    ;       that structure and an ordered offset table is constructed
                                07F3    ;       with pointers to the actual symbol table entries.
                                07F3    ;       Only one offset table may be in use at any time.
                                07F3    ;       A later specification of another structure in a FORMAT
                                07F3    ;       command overrides the current specification.
                                07F3    ;
                                07F3    ; Inputs:
                                07F3    ;
                                07F3    ;       AP = TPARSE block (TPA$L_NUMBER contains the address)
                                07F3    ;       OPTIONS = Non-zero if structure specified, else zero.
                                07F3    ;       STRUCTURE = Quadword descriptor of structure string.
                                07F3    ;
                                07F3    ;---
                                07F3          .enabl  lsb
                                07F3
                                07F3    typetab:
     43 54 4C 57 42 51 47 41    07F3          .ascii  "AGQBWLTC"               ; Valid types for FORMAT
                    00000008    07FB    typelen = .-typetab
                                07FB
                        0FFC    07FB          .entry  FORMAT,-
                                07FD                  ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                                07FD
 00000000'EF    1C AC     D0    07FD          movl    tpa$l_number(ap),address ; Set current location
       00000000'EF        D5    0805          tstl    options                  ; Any new structure specified?
                  03      13    080B          beql    1$                       ; If so, check the string
                0OAB      31    080D          brw     10$
                                0810    ;
                                0810    ;       Obtain the structure name from the block type
                                0810    ;
    50    0A    1C AC     C1    0810    1$:   addl3   tpa$l_number(ap),#irp$b_type,r0
                                0815          getmem  (r0)                     ; Get the block type
               03 50      E8    081E          blbs    r0,2$                    ; Branch if cannot read
                0093      31    0821          brw     80$
                                0824    ;
                                0824    ;       Check for subtype
                                0824    ;
         60 8F    51      91    0824    2$:   cmpb    r1, #dyn$c_subtype       ; check to see if in subtype region
                  4C      1F    0828          blssu   3$                       ; less than, not in region
         80 8F    51      91    082A          cmpb    r1, #dyn$c_special       ; max limit of subtype region
                  46      1E    082E          bgequ   3$                       ; greater or equal, beyond region
   52    51    08 08      EF    0830          extzv   #8,#8,r1,r2              ; extract subtype
               51 51      9A    0835          movzbl  r1,r1                    ; zero-extend the field
          51   60 8F      82    0838          subb2   #dyn$c_subtype, r1       ; yields offset into main type table
   51     00000000'EF41   B0    083C          movw    dyn_mainptr[r1], r1      ; pick up start of offsets to subtypes
                  52      D7    0844          decl    r2                       ; decrement subtype by 1( offsets start at 0
               52 51      C0    0846          addl2   r1,r2                    ; actual offset in table of offsets
   51     00000000'EF42   B0    0849          movw    dyn_subptr[r2], r1       ; pick up offset in name table
   52     00000000'EF41   9E    0851          movab   dyn_tab[r1], r2          ; address of name
               51 62      9B    0859          movzbw  (r2), r1                 ; put length in register 1
 01 A2    51    5F 8F     3A    085C          locc    #95, r1, 1(r2)           ; check for underscore character
```

```
                          31  13  0862        beql    4$                      ; eql, no underscore char
   00000020'EF       50   01  C3  0864        subl3   #1, r0, structure       ; set length of name
     00000024'EF     01   A1  9E  086C        movab   1(r1), structure+4      ; set address of name
                          45  11  0874        brb     10$
                              0876   ;
                              0876   ; No sub type
                              0876   ;
                     51   51  9A  0876   3$:  movzbl  r1,r1                   ; Zero-extend the field
                          2A  13  0879        beql    5$                      ; Branch if invalid
           52    00000000'EF41  98  087B      cvtbl   dyn_map[r1],r2          ; Pick up symbol info
                          20  19  0883        blss    5$                      ; Branch if invalid
           52    00000000'EF41  3C  0885      movzwl  dyn_ptr[r1],r2          ; Pick up offset in name table
           52    00000000'EF42  9E  088D      movab   dyn_tab[r2],r2          ; Address of name
        00000020'EF       82  9A  0895   4$:  movzbl  (r2)+,structure         ; Set length of name
        00000024'EF       52  D0  089C        movl    r2,structure+4          ; Set address of name
                          16  11  08A3        brb     10$
                              08A5
                              08A5   5$:      signal  0,invblktyp             ; Signal invalid block type
                     50   01  D0  08B7   80$: movl    #1,r0
                              04  08BA        ret
                              08BB   ;
                              08BB   ;        Check string specified in structure descriptor
                              08BB   ;
           56    00000020'EF  7D  08BB   10$: movq    structure,r6            ; Get descriptor of string
                     09   56  B1  08C2        cmpw    r6,#9                   ; Check if reasonable length
                     DE   14      08C5        bgtr    5$                      ; Error if not
                              08C7   ;
                              08C7   ;        Construct table of symbol table pointers for this structure,
                              08C7   ;        sorted in ascending value order.
                              08C7   ;
     F868 CF     67   03  29  08C7        cmpc3   #prefix_len,(r7),ucb_prefix     ; check to see if the structure is a
                          2B  13  08CD        beql    30$                     ; branch if the ucb - format special
                        030C  30  08CF        bsbw    sort_offsets            ; Construct offset table
           56    00000018'EF  7D  08D2   25$: movq    offset_table,r6         ; Get starting/ending table addresses
                     57   56  D1  08D9        cmpl    r6,r7                   ; Anything in table?
                          3D  12  08DC        bneq    50$                     ; Branch if table non-empty
              00000020'EF     9F  08DE        pushab  structure               ; Address of structure name
                              08E4            signal  1,nosymbols             ; Signal no symbols found
                     50   01  D0  08F6        movl    #1,r0
                              04  08F9        ret
                              08FA
           55    00000028'EF  DE  08FA   30$: moval   ucb_struct,r5           ; move address of local ucb
                              0901            getmem  @tpa$l_number(ap),(r5),#ucb$c_length    ; attempt to read it
                       A1 50  E9  0913        blbc    r0,80$                  ; branch if cannot read
                        011C  30  0916        bsbw    ucb_sort_offsets        ; special case - ucb structure
                          B7  11  0919        brb     25$                     ; rejoin common code
                              091B   ;
                              091B   ;        Format the data structure
                              091B   ;
              58      00 B6   D0  091B   50$: movl    @(r6),r8                ; Pick up first offset
                          02  15  091F        bleq    60$                     ; Negative/zero okay
                          58  D4  0921        clrl    r8                      ; If positive, start with zero
                     57   56  D1  0923   60$: cmpl    r6,r7                   ; End of table yet?
                          3B  1E  0926        bgequ   90$                     ; Branch if so
                     59   86  D0  0928        movl    (r6)+,r9                ; Get address of symbol entry
              00000000'EF     D4  092B        clrl    line_count              ; No page ejects during display
              5A      04 A9   9E  0931        movab   sym$t_symbol(r9),r10    ; Address of symbol to display
```

```
51   00000020'EF    01   C1  0935        addl3   #1,structure,r1               ; Character index to data type
         51   05 A941   90  093D        movb    sym$t_symbol+1(r9)[r1],r1     ; Get data type character
              58    69   D1  0942        cmpl    sym$l_value(r9),r8            ; Check against current offset
                    20   19  0945        blss    100$                          ; Process duplicate symbols
                    31   14  0947        bgtr    200$                          ; Process un-named data
   FEA4 CF    08    51   3A  0949        locc    r1,#typelen,typetab           ; Index into type table
                             094F        case    r0,type=b,limit=#1,-          ; Case on data type
                             094F                <100$,110$,120$,130$,140$,120$,120$,120$>
              50    01   D0  0963 90$:   movl    #1,r0
                         04  0966        ret
                             0967
                             0967 ;
                             0967 ;      Process duplicate symbol with the same offset by
                             0967 ;      printing the symbol without any data next to it
                             0967 ;
                 04 A9   9F  0967 100$:  pushab  sym$t_symbol(r9)              ; Address of symbol name
                             096A        print   1,<             !24AC>        ; Print only symbol name
                    AA   11  0977        brb     60$
                             0979 ;
                             0979 ;      Process un-named data
                             0979 ;
                         00' 0979 null:  .ascic  ''                            ; Null string
                         00  0979
                             097A
              56    04   C2  097A 200$:  subl    #4,r6                         ; Back up over skipped symbol
         5A      F9 AF   9E  097D        movab   b^null,r10                    ; Leave blank for symbol
       54    69     58   C3  0981        subl3   r8,sym$l_value(r9),r4         ; Length of unnamed bytes
                    50   11  0985        brb     201$
                             0987 ;
                             0987 ;      Process ASCIC text data
                             0987 ;
52   00000000'EF         9E  0987 110$:  movab   buffer,r2                     ; Address of work buffer
                             098E        getmem  @tpa$l_number(ap)[r8],(r2),#32 ; Read ASCIC string
                 C3 50   E9  099D        blbc    r0,90$                        ; If cannot read, skip this
              20    62   91  09A0        cmpb    (r2),#32                      ; Check if reasonable length
                    22   1A  09A3        bgtru   115$                          ; If not, process as un-named
                 01 A2   9F  09A5        pushab  1(r2)                         ; Address of ASCIC string
              7E    62   9A  09A8        movzbl  (r2),-(sp)                    ; Length of string
                 04 A9   9F  09AB        pushab  sym$t_symbol(r9)              ; Address of symbol name
               1C BC48   9F  09AE        pushab  @tpa$l_number(ap)[r8]         ; Address of data
                             09B2        print   4,<!XL   !24AC"!AF">
              50    62   9A  09BF        movzbl  (r2),r0                       ; Get length of string
              58    50   C0  09C2        addl    r0,r8                         ; Add string length
                    58   D6  09C5        incl    r8                            ; plus one for length byte
                  FF59   31  09C7 115$:  brw     60$
                             09CA ;
                             09CA ;      Process longword data
                             09CA ;
              54    04   D0  09CA 120$:  movl    #4,r4                         ; Length to display
                    08   11  09CD        brb     201$
                             09CF ;
                             09CF ;      Process word data
                             09CF ;
              54    02   D0  09CF 130$:  movl    #2,r4                         ; Length to display
                    03   11  09D2        brb     201$
                             09D4 ;
                             09D4 ;      Process byte data
```

```
                        09D4   ;
         54    01   D0  09D4   140$:   movl    #1,r4                        ; Length to display
                        09D7   ;
                        09D7   ;       Process any type of data
                        09D7   ;
53   58  02    00   EF  09D7   201$:   extzv   #0,#2,r8,r3                  ; Extract longword alignment
     53  04    53   C3  09DC           subl3   r3,#4,r3                     ; Bytes to display
         55    53   D0  09E0           movl    r3,r5                        ; Save for later
         54    53   D1  09E3           cmpl    r3,r4                        ; More than enough?
               03   15  09E6           bleq    205$                         ; Branch if ok
         53    54   D0  09E8           movl    r4,r3                        ; If so, only print that much
                        09EB   205$:   getmem  @tpa$l_number(ap)[r8]        ; Read remaining bytes
            38 50   E9  09F6           blbc    r0,230$                      ; Branch if cannot read
         50    53   D0  09F9           movl    r3,r0                        ; Loop count
         7E    51   9A  09FC   210$:   movzbl  r1,-(sp)                     ; Display the byte
51   51     F8 8F   78  09FF           ashl    #-8,r1,r1                    ; Shift to next byte
               50   D7  0A04           decl    r0                           ; Decrement byte count
               F4   14  0A06           bgtr    210$                         ; Loop until done
               53   DD  0A08           pushl   r3                           ; Number of bytes valid
         55    53   C2  0A0A           subl    r3,r5                        ; Amount truncated from left
     7E  55    01   78  0A0D           ashl    #1,r5,-(sp)                  ; 2 positions/byte
               5A   DD  0A11           pushl   r10                          ; Address of symbol name
         1C  BC48   9F  0A13           pushab  @tpa$l_number(ap)[r8]        ; Address of data
                        0A17           print   4,<!XL   !24AC!#* !#(XB)>
     50  53    02   78  0A24           ashl    #2,r3,r0                     ; Bytes still on stack
         5E    50   C0  0A28           addl    r0,sp                        ; Clean up stack
         58    53   C0  0A2B   220$:   addl    r3,r8                        ; Skip data printed
             FEF2   31  0A2E           brw     60$
         50    01   D0  0A31   230$:   movl    #1,r0
                    04  0A34           ret
                        0A35
                        0A35           .dsabl  lsb
```

```
                            0A35         .sbttl  ucb_sort_offsets, sort offsets for the ucb
                            0A35  ;---
                            0A35  ;
                            0A35  ;      ucb_sort_offsets
                            0A35  ;
                            0A35  ;      The ucb is an exception to formatting a data structure since
                            0A35  ;      we do not want to display all its symbols for a given device.
                            0A35  ;      The ucb contains extensions to the standard ucb for many different
                            0A35  ;      devices.  This routine will select those symbols which are part of
                            0A35  ;      the standard ucb along with only the extension symbols which pertain
                            0A35  ;      to this device.  A table of these symbols, sorted by value, is
                            0A35  ;      constructed.  This table will then be used by the FORMAT routine to
                            0A35  ;      display the contents of the ucb.
                            0A35  ;
                            0A35  ;
                            0A35  ; Inputs:
                            0A35  ;
                            0A35  ;      r5    = local copy of the ucb
                            0A35  ;      r6/r7 = Descriptor of structure prefix string
                            0A35  ;
                            0A35  ; Outputs:
                            0A35  ;
                            0A35  ;      offset_table = Pointer to array of addresses into symbol table entries
                            0A35  ;                             sorted in numeric order.
                            0A35  ;---
                            0A35  ucb_sort_offsets:
                            0A35  ;
                            0A35  ;      Allocate buffer to hold ordered offset table
                            0A35  ;
52   00000018'EF      7E    0A35         movaq   offset_table,r2          ; Address of quadword
     00000800 8F      DD    0A3C         pushl   #<256*4>*2               ; Maximum of 256 offsets, 2 tables:
                            0A42                                          ; 256 pointers to sym entries
                            0A42                                          ; 256 symbol values, to optimize sorting
                            0A42                                          ; (minimizes page faults into symtable)
      F98D CF    01   FB    0A42         calls   #1,allocate              ; Allocate the storage
           58    51   D0    0A47         movl    r1,r8                    ; Save address of table
           62    58   D0    0A4A         movl    r8,(r2)
        04 A2    58   D0    0A4D         movl    r8,4(r2)                 ; Mark table empty
           08         11    0A51         brb     10$
                            0A53
0000001C'EF      58   D0    0A53  5$:    movl    r8,offset_table+4        ; mark last valid table entry
                      05    0A5A         rsb                              ; exit with newly constructed table
                            0A5B  ;
                            0A5B  ;      Now translate the device class to a routine address.  This routine
                            0A5B  ;      will decide if the symbol is valid for this device if the symbol
                            0A5B  ;      is part of the ucb extension region.  If the symbol lies in the
                            0A5B  ;      standard ucb it will be inserted into the table.
                            0A5B  ;
53   000006F3'EF      9E    0A5B  10$:   movab   devclass_table,r3              ; translation table
        52    40 A5   9A    0A62         movzbl  ucb$b_devclass(r5),r2          ; translate device class to routine
     00000D04'EF      16    0A66         jsb     translate_address              ; get address corresponding to devcl
           03         12    0A6C         bneq    15$                            ; translation valid
           50    00   D0    0A6E         movl    #0,r0                          ; zero the address
0000072B'EF      50   D0    0A71  15$:   movl    r0,device_rout                 ; move to safe grounds-use later
                            0A78  ;
                            0A78  ;      Find the highest node in the alpha tree containing the required
                            0A78  ;      prefix.  This is the parent of all other nodes with the prefix,
```

```
                         0A78       ;       so then traverse the subtree rooted in this node.  Use a simple
                         0A78       ;       insertion sort to get the symbols in increasing order of value.
                         0A78       ;
   59 00000010'EF   D0   0A78               movl    alpfl,r9                        ; get root of alpha table
           FC69 CF  9F   0A7F               pushab  alp_traverse                    ; push prefix-scan routine
                9E  16   0A83       20$:    jsb     @(sp)+                          ; get next symbol
             CB 50  E9   0A85               blbc    r0, 5$                          ; J if end of tree reached
   53     05 AA46   9E   0A88               movab   sym$t_symbol+1(r10)[r6], r3     ; setup pointer to end of prefix
      52    04 AA   9A   0A8D               movzbl  sym$t_symbol(r10),r2            ; Get length of symbol
         52    56   C2   0A91               subl    r6,r2                           ; Length left after prefix
         04    52   D1   0A94               cmpl    r2,#4                           ; Must be at least 4 chars left
               EA   19   0A97               blss    20$                             ; If not, no match
         24    83   91   0A99               cmpb    (r3)+,#^a'$'                    ; Must be followed by $
               07   13   0A9C               beql    25$                             ; Branch if valid
   5F 8F    FF A3   91   0A9E               cmpb    -1(r3),#^a'_'                   ; Allow underscore as alternate
               DE   12   0AA3               bneq    20$                             ; Branch if not valid
         54    53   D0   0AA5       25$:    movl    r3,r4                           ; Save pointer
 FD45 CF    08 83   3A   0AA8               locc    (r3)+,#typelen,typetab          ; Check if valid type
               D3   13   0AAE               beql    20$                             ; Branch if not a valid type
      43 8F    61   91   0AB0               cmpb    (r1),#^a/C/                     ; check if constant symbol
               0E   12   0AB4               bneq    26$                             ; if not, continue processing
     6A    08 A5    B1   0AB6               cmpw    ucb$w_size(r5),sym$l_value(r10) ; see if length symbol we want
               C7   12   0ABA               bneq    20$                             ; if not, ignore it
      0000072B'EF   D5   0ABC               tstl    device_rout                     ; if no extension, this must be
               0F   13   0AC2               beql    30$                             ; length of ucb, so insert
      5F 8F    83   91   0AC4       26$:    cmpb    (r3)+,#^a'_'                    ; Must be followed by underscore
               B9   12   0AC8               bneq    20$                             ; Branch if not
                         0ACA       ;
                         0ACA       ;       It is now clear that the symbol is valid.  The question now remains
                         0ACA       ;       is it a valid ucb symbol.  If it lies in the standard ucb range
                         0ACA       ;       insert it into the table, no questions asked.  If it is in the
                         0ACA       ;       extension region, check to see if it is part of this device's
                         0ACA       ;       symbols.  If it is, insert into table, otherwise ignore it.
                         0ACA       ;
 00000090 8F    6A  D1   0ACA               cmpl    sym$l_value(r10),#ucb$c_length  ; part of standard ucb?
               05   18   0AD1               bgeq    40$                             ; grt, part of extension
             0190   30   0AD3       30$:    bsbw    table_insert                    ; insert symbol and value into table
               AB   11   0AD6               brb     20$                             ; get next symbol
                         0AD8
                         0AD8       ;       Symbol lies in the extension region
                         0AD8       ;
                         0AD8       40$:
      0000072B'EF   D5   0AD8               tstl    device_rout                     ; check for valid address
               0B   13   0ADE               beql    45$                             ; no device extension
      0000072B'EF   DD   0AE0               pushl   device_rout                     ; device dependent routine
               9E   16   0AE6               jsb     @(sp)+                          ; check if valid ucb symbol
            E8 50   E8   0AE8               blbs    r0,30$                          ; valid, so insert into table
02 38 A5  00000000'8F  E0  0AEB     45$:    bbs     #dev$v_elg,ucb$l_devchar(r5), 50$   ; error logging symbol?
               8D   11   0AF4               brb     20$                             ; not valid, so ignore it
                         0AF6       ;
                         0AF6       ;       Error Logging Device
                         0AF6       ;
               53   DD   0AF6       50$:    pushl   r3                              ; address of symbol
               52   DD   0AF8               pushl   r2                              ; length of symbol
      000007AD'EF   9F   0AFA               pushab  errorlog_ext                    ; table of erl symbols
   00000BBA'EF 03   FB   0B00               calls   #3,valid_ucbsymbol              ; is this symbol in table
            C9 50   E8   0B07               blbs    r0,30$                          ; yes, insert in table
```

```
             FF76  31  0B0A          brw     20$                              ; no , ignore it
                       0B0D  ;
                       0B0D  ;       Disks and Tapes
                       0B0D  ;
                       0B0D  ucb_dt:
               53  DD  0B0D          pushl   r3                               ; address of symbol
               52  DD  0B0F          pushl   r2                               ; length of symbol
        00000813'EF  9F  0B11          pushab  disks_tapes                      ; table of valid symbols
   00000BBA'EF  03  FB  0B17          calls   #3,valid_ucbsymbol               ; is there a match
            22 50  E8  0B1E          blbs    r0,20$                           ; if set, symbol valid
               53  DD  0B21          pushl   r3                               ; check tables of mscp or local
               52  DD  0B23          pushl   r2                               ; length of symbol
08 3C A5  00000000'8F  E1  0B25          bbc     #dev$v_mscp,ucb$l_devchar2(r5),5$  ; mscp device?
        0000089A'EF  9F  0B2E          pushab  mscp_ext                         ; table of mscp symbols
               06  11  0B34          brb     10$
        0000092B'EF  9F  0B36  5$:     pushab  ldisk_ext                        ; table of local disk symbols
   00000BBA'EF  03  FB  0B3C  10$:    calls   #3,valid_ucbsymbol                      ; do we want this symbol?
                       0B43  20$:
                   05  0B43          rsb
                       0B44
                       0B44  ;       Terminal Device
                       0B44  ;
                       0B44  ucb_term:
               53  DD  0B44          pushl   r3                               ; address of symbol
               52  DD  0B46          pushl   r2                               ; length of symbol
08 3C A5  00000000'8F  E0  0B48          bbs     #dev$v_rtt,ucb$l_devchar2(r5),5$        ; set, remote terminal
        0000075A'EF  9F  0B51          pushab  lterm_ext                        ; table of valid prefixes for
               06  11  0B57          brb     10$                              ; local terminals
        0000077C'EF  9F  0B59  5$:     pushab  rterm_ext                        ; table of valid prefixes for remote
   00000BBA'EF  03  FB  0B5F  10$:    calls   #3,valid_ucbsymbol               ; is there a match
                   05  0B66          rsb
                       0B67
                       0B67  ;       Network Device
                       0B67  ;
                       0B67  ucb_bus:
               53  DD  0B67          pushl   r3                               ; address of symbol
               52  DD  0B69          pushl   r2                               ; length of symbol
        0000079F'EF  9F  0B6B          pushab  bus_ext                          ; table of valid symbols
   00000BBA'EF  03  FB  0B71          calls   #3,valid_ucbsymbol               ; is there a match
                   05  0B78          rsb
                       0B79
                       0B79  ;       Mailbox device
                       0B79  ;
                       0B79  ucb_mail:
               53  DD  0B79          pushl   r3                               ; address of symbol
               52  DD  0B7B          pushl   r2                               ; length of symbol
        0000073E'EF  9F  0B7D          pushab  mailbx_ext                       ; table of valid prefixes
   00000BBA'EF  03  FB  0B83          calls   #3,valid_ucbsymbol               ; is there a match
            1A 50  E8  0B8A          blbs    r0,10$                           ; if set, found match
11 38 A5  00000000'8F  E1  0B8D          bbc     #dev$v_net,ucb$l_devchar(r5),10$  ; check for network mailbox
               53  DD  0B96          pushl   r3                               ; address of symbol
               52  DD  0B98          pushl   r2                               ; push length of symbol
        0000074C'EF  9F  0B9A          pushab  netmbx_ext                       ; network mailbox prefix
   00000BBA'EF  03  FB  0BA0          calls   #3,valid_ucbsymbol               ; check validity
                   05  0BA7  10$:    rsb
                       0BA8
                       0BA8  ;       Journaling Device
```

```
                     0BA8  1774 ;
                     0BA8  1775 ucb_journal:
            53   DD  0BA8  1776         pushl   r3                      ; address of symbol
            52   DD  0BAA  1777         pushl   r2                      ; length of symbol
    0000072F'EF  9F  0BAC  1778         pushab  journal_ext             ; table of valid symbols
00000BBA'EF  03  FB  0BB2  1779         calls   #3,valid_ucbsymbol      ; is there a match
            05       0BB9  1780         rsb
```

```
                              0BBA        .sbttl  valid_ucbsymbol, valid ucb symbol for this device
                              0BBA   ;---
                              0BBA   ;
                              0BBA   ;        valid_ucbsymbol
                              0BBA   ;
                              0BBA   ;        This routine checks to see if a given symbol is in the
                              0BBA   ;        table of symbols that correspond to this device.
                              0BBA   ;
                              0BBA   ; Inputs:
                              0BBA   ;
                              0BBA   ;        4(ap)  = address of table of symbols for this device
                              0BBA   ;        8(ap)  = length of symbol minus prefix
                              0BBA   ;        12(ap) = address of symbol to check for validity
                              0BBA   ;
                              0BBA   ; Outputs:
                              0BBA   ;
                              0BBA   ;        r0 = 1 if a match was found, 0 if no match
                              0BBA   ;        Preserves all registers
                              0BBA   ;
                              0BBA   valid_ucbsymbol:
                        001C  0BBA        .word   ^m<r2,r3,r4>
          54    04 AC     D0  0BBC        movl    4(ap),r4                    ; address of extension symbols
          08 AC    03     C2  0BC0        subl2   #3,8(ap)                    ; subtract $x_ from length
04 B4     0C BC    64     29  0BC4   5$:  cmpc3   (r4),@12(ap),@4(r4)         ; match?
                   0E     13  0BCA        beql    10$                         ; equal, yes
          54       64     A0  0BCC        addw2   (r4),r4                     ; point at next symbol by adding the length
          54       08     C0  0BCF        addl2   #8,r4                       ; of symbol + length of descriptor (8)
          00       64     D1  0BD2        cmpl    (r4),#0                     ; end of table?
                   ED     12  0BD5        bneq    5$                          ; not equal, no
                   50     D4  0BD7        clrl    r0                          ; no match found
                          04  0BD9        ret                                 ; return
                              0BDA   10$:
          50       01     D0  0BDA        movl    #1,r0                       ; match found
                          04  0BDD        ret                                 ; return
```

```
                          0BDE        .sbttl  sort_offsets, sort offsets for a given structure
                          0BDE   ;---
                          0BDE   ;
                          0BDE   ;        sort_offsets
                          0BDE   ;
                          0BDE   ;        This routine constructs a table of symbols for a given
                          0BDE   ;        structure, sorted by value.  The table is then used by
                          0BDE   ;        the FORMAT command to display the contents of a structure.
                          0BDE   ;
                          0BDE   ; Inputs:
                          0BDE   ;
                          0BDE   ;        r6/r7 = Descriptor of structure prefix string
                          0BDE   ;
                          0BDE   ; Outputs:
                          0BDE   ;
                          0BDE   ;        offset_table = Pointer to array of addresses into symbol table entries
                          0BDE   ;                                sorted in numeric order.
                          0BDE   ;---
                          0BDE   sort_offsets:
                          0BDE   ;
                          0BDE   ;        Allocate buffer to hold ordered offset table
                          0BDE   ;
52   00000018'EF    7E    0BDE        movaq   offset_table,r2          ; Address of quadword
           58  62   D0    0BE5        movl    (r2),r8                  ; Address of previous table
               17   12    0BE8        bneq    10$                      ; Branch if already exists
     00000800 8F    DD    0BEA        pushl   #<256*4>*2               ; Maximum of 320 offsets, 2 tables:
                          0BF0                                         ; 320 pointers to sym entries
                          0BF0                                         ; 320 symbol values, to optimize sorting
                          0BF0                                         ; (minimizes page faults into symtable)
     F7DF CF   01   FB    0BF0        calls   #1,allocate              ; Allocate the storage
           58  51   D0    0BF5        movl    r1,r8                    ; Save address of table
           62  58   D0    0BF8        movl    r8,(r2)
        04 A2  58   D0    0BFB        movl    r8,4(r2)                 ; Mark table empty
               22   11    0BFF        brb     16$
                          0C01   ;
                          0C01   ;        If table already holds offset symbols for the desired
                          0C01   ;        structure, skip searching the entire symbol table.
                          0C01   ;
        04 A2  58   D1    0C01   10$:  cmpl    r8,4(r2)                 ; Check if any entries
               1C   1E    0C05        bgequ   16$                      ; Skip if none
           51  68   D0    0C07        movl    (r8),r1                  ; Get pointer to first symbol
     50    04  A1   9A    0C0A        movzbl  sym$t_symbol(r1),r0      ; Get length of first symbol
           56  50   D1    0C0E        cmpl    r0,r6                    ; Greater than prefix length?
               10   1B    0C11        blequ   16$                      ; Skip compare if so
 05 A1     67  56   29    0C13        cmpc3   r6,(r7),sym$t_symbol+1(r1) ; Same prefix?
               09   12    0C18        bneq    16$                      ; if not, create new structure
                    05    0C1A        rsb                              ; else just go use it
                          0C1B
0000001C'EF    58   D0    0C1B   15$:  movl    r8,offset_table+4        ; mark last valid table entry
                    05    0C22        rsb                              ; exit with newly constructed table
                          0C23   ;
                          0C23   ;        Find the highest node in the alpha tree containing the required
                          0C23   ;        prefix.  This is the parent of all other nodes with the prefix,
                          0C23   ;        so then traverse the subtree rooted in this node.  Use a simple
                          0C23   ;        insertion sort to get the symbols in increasing order of value.
                          0C23   ;
59   00000010'EF    D0    0C23   16$:  movl    alpfl,r9                         ; get root of alpha table
```

```
          FABE CF   9F  0C2A  1874       pushab  alp_traverse                    ; push prefix-scan routine
               9E   16  0C2E  1875 20$:  jsb     @(sp)+                          ; get next symbol
            E8 50   E9  0C30  1876       blbc    r0, 15$                         ; J if end of tree reached
     53   05 AA46   9E  0C33  1877       movab   sym$t_symbol+1(r10)[r6], r3     ; setup pointer to end of prefix
        52   04 AA  9A  0C38  1878       movzbl  sym$t_symbol(r10),r2            ; Get length of symbol
           52   56  C2  0C3C  1879       subl    r6,r2                           ; Length left after prefix
           04   52  D1  0C3F  1880       cmpl    r2,#4                           ; Must be at least 4 chars left
                EA  19  0C42  1881       blss    20$                             ; If not, no match
           24   83  91  0C44  1882       cmpb    (r3)+,#^a'$'                    ; Must be followed by $
                07  13  0C47  1883       beql    24$                             ; Branch if valid
     5F 8F   FF A3  91  0C49  1884       cmpb    -1(r3),#^a'_'                   ; Allow underscore as alternate
                DE  12  0C4E  1885       bneq    20$                             ; Branch if not valid
           54   53  D0  0C50  1886 24$:  movl    r3,r4                           ; Save pointer
 FB9A CF   08   83  3A  0C53  1887       locc    (r3)+,#typelen,typetab          ; Check if valid type
                D3  13  0C59  1888       beql    20$                             ; Branch if not a valid type
        5F 8F   83  91  0C5B  1889       cmpb    (r3)+,#^a'_'                    ; Must be followed by underscore
                CD  12  0C5F  1890       bneq    20$                             ; Branch if not
              0002  30  0C61  1891       bsbw    table_insert                    ; insert symbol and value into table
                C8  11  0C64  1892       brb     20$                             ; get next symbol
```

```
                              0C66        .sbttl  table_insert, insert valid symbol into ordered table
                              0C66  ;---
                              0C66  ;
                              0C66  ;       table_insert
                              0C66  ;
                              0C66  ;       We have a valid symbol.  Search up value table to find the right
                              0C66  ;       place to insert, then shuffle up all the following entries in
                              0C66  ;       both tables. There is one table for the symbol text and one for
                              0C66  ;       the value of the symbol.  The two tables are contiguous.  The
                              0C66  ;       table for the symbol text is pointed to by the variable, offset_table.
                              0C66  ;       The table of values is a fixed offset from the offset_table.
                              0C66  ;
                              0C66  ; Inputs:
                              0C66  ;
                              0C66  ;       R2      = Length of symbol minus the length of the prefix
                              0C66  ;       R4      = Pointer to type of symbol (i.e. B,L,C)
                              0C66  ;       R8      = Address of last symbol in the table
                              0C66  ;       R10     = Address of symbol block
                              0C66  ;
                              0C66  ; Outputs:
                              0C66  ;
                              0C66  ;       R8      = Updated address of last symbol in the table
                              0C66  ;       Destroys registers, r1,r3, and r11
                              0C66  ;
                              0C66  table_insert:
              5B    52   D0   0C66          movl    r2,r11                  ; save length of symbol
      51  00000018'EF    D0   0C69          movl    offset_table,r1         ; get start of pointer array
         52   0400 C1    9E   0C70          movab   256*4(r1),r2            ; get start of value array
              53    6A   D0   0C75          movl    sym$l_value(r10),r3     ; save symbol value
                              0C78  ;
                              0C78  ; Check to see if character is a constant.  The only constant to be
                              0C78  ; displayed is the length character.  The UCB is a special structure
                              0C78  ; since it can be of variable length.  This is a temporary fix for the UCB.
                              0C78  ;
           43 8F    64   91   0C78          cmpb    (r4),#^a/C/             ; is it a xxx$C_yyy symbol?
                    14   12   0C7C          bneq    10$                     ; no, skip on
                    0E   BB   0C7E          pushr   #^m<r1,r2,r3>           ; save registers
 64   5B   F4A9 CF  07   39   0C80          matchc  #l_len,length_sym,r11,(r4)    ; if length symbol, o.k. to display
                    34   12   0C87          bneq    40$                     ; otherwise forget we saw this symbol
                    0E   BA   0C89          popr    #^m<r1,r2,r3>           ; restore registers
      53   70000000 8F   C8   0C8B          bisl2   #^x70000000,r3          ; make large number so end of table
                              0C92  ;
              58    51   D1   0C92  10$:    cmpl    r1,r8                   ; reached end of table yet?
                    0B   1E   0C95          bgequ   30$                     ; J if so
              82    53   D1   0C97          cmpl    r3,(r2)+                ; found right place yet?
                    04   19   0C9A          blss    20$                     ; J if so (signed branch for neg off)
                    81   D5   0C9C          tstl    (r1)+                   ; else advance STRUC_TBL pointer too
                    F2   11   0C9E          brb     10$                     ; and loop
                              0CA0  20$:                                    ; found place to insert!
                    72   D5   0CA0          tstl    -(r2)                   ; back off VAL_TBL pointer
              5B    62   D0   0CA2  30$:    movl    (r2),r11                ; save existing table entries
              54    61   D0   0CA5          movl    (r1),r4                 ; ...
              81    5A   D0   0CA8          movl    r10,(r1)+               ; save symbol entry pointer
              82    53   D0   0CAB          movl    r3,(r2)+                ; and value
              53    5B   D0   0CAE          movl    r11,r3                  ; get values to temp registers
              5A    54   D0   0CB1          movl    r4,r10                  ; ...
              58    51   D1   0CB4          cmpl    r1,r8                   ; reached end of table yet?
```

```
             E9   1B  0CB7  1951          blequ   30$                     ; loop if not
     58      51   D0  0CB9  1952          movl    r1,r8                   ; save new end pointer
                  05  0CBC  1953          rsb
                      0CBD  1954
             0E   BA  0CBD  1955  40$:    popr    #^m<r1,r2,r3>           ; restore registers
                  05  0CBF  1956          rsb                             ; get next symbol
                      0CC0  1957
                      0CC0  1958
```

```
                    0CC0        .sbttl  translate_bits, translate bit mask to alpha string
                    0CC0  ;---
                    0CC0  ;
                    0CC0  ;     translate_bits
                    0CC0  ;
                    0CC0  ;     This routine translates a given bit mask into a string
                    0CC0  ;     of the names of each of the bits set in the mask, each
                    0CC0  ;     separated by delimiters.
                    0CC0  ;
                    0CC0  ; Inputs:
                    0CC0  ;
                    0CC0  ;     4(ap)   = address of bit definition table
                    0CC0  ;     8(ap)   = bit mask to translate
                    0CC0  ;     12(ap)  = descriptor of output buffer
                    0CC0  ;
                    0CC0  ;     The bit definition table is formatted as follows:
                    0CC0  ;
                    0CC0  ;     value1,address1
                    0CC0  ;     value2,address2
                    0CC0  ;     value3,address3
                    0CC0  ;     etc.
                    0CC0  ;     -1,-1
                    0CC0  ;
                    0CC0  ;     where "value" is the bit number and "address" is the
                    0CC0  ;     address of the counted ascii name string associated
                    0CC0  ;     with that bit.
                    0CC0  ;
                    0CC0  ; Outputs:
                    0CC0  ;
                    0CC0  ;     The output buffer contains the name string.
                    0CC0  ;     12(ap)  = descriptor of result string
                    0CC0  ;
                    0CC0  ; Calling sequence:
                    0CC0  ;
                    0CC0  ;     alloc   80,r2                   ; allocate output buffer
                    0CC0  ;     movzwl  ucb$w_sts(r3),-(sp)     ; bit mask to translate
                    0CC0  ;     pushab  unit_status             ; address of definition table
                    0CC0  ;     calls   #2,translate_bits       ; translate bits into names
                    0CC0  ;
                    0CC0  ;     Notice that the output buffer descriptor is not included
                    0CC0  ;     in the argument count so that it stays on the stack after
                    0CC0  ;     the routine is finished.
                    0CC0  ;
                    0CC0  ;---
                    0CC0
                    0CC0  translate_bits::
              007C  0CC0        .word   ^m<r2,r3,r4,r5,r6>
                    0CC2
          52    D4  0CC2        clrl    r2                      ; starting bit number
  53   04 AC    7D  0CC4        movq    4(ap),r3                ; get table address & bit mask
  55   0C AC    7D  0CC8        movq    12(ap),r5               ; output buffer length,pointer
                    0CCC  10$:
1F 54     52    E1  0CCC        bbc     r2,r4,50$               ; skip if bit not set
                    0CD0
          32    10  0CD0        bsbb    translate_address       ; translate bit number to str. addr.
          1B    13  0CD2        beql    50$                     ; branch if no tranlation
                    0CD4
```

```
            51    80    9A  0CD4         movzbl  (r0)+,r1              ; get length of string
            55    51    C2  0CD7         subl    r1,r5                 ; decrement buffer space left
                  1A    19  0CDA         blss    90$                   ; branch if not enough room
         10 AC    56    D1  0CDC         cmpl    r6,16(ap)             ; check if first name in string
                  07    13  0CE0         beql    35$                   ; branch if so
                  55    D7  0CE2         decl    r5                    ; decrement buffer space left
                  10    19  0CE4         blss    90$                   ; branch if not enough room
            86    2C    90  0CE6         movb    #^a',',(r6)+          ; insert comma in between names
                            0CE9   35$:
            86    80    90  0CE9         movb    (r0)+,(r6)+           ; concatenate name to string
               FA 51    F5  0CEC         sobgtr  r1,35$
                            0CEF   50$:
                  52    D6  0CEF         incl    r2                    ; next bit in mask
            1F    52    D1  0CF1         cmpl    r2,#31                ; are we done with longword?
                  D6    15  0CF4         bleq    10$                   ; loop if not
                            0CF6
                            0CF6   90$:
 0C AC    56   10 AC    C3  0CF6         subl3   16(ap),r6,12(ap)      ; length of result string
                            0CFC         status  success
                        04  0D03         ret
```

```
                  0D04       .SBTTL  translate_address, translate value to an address
                  0D04  ;---
                  0D04  ;
                  0D04  ;       translate_address
                  0D04  ;
                  0D04  ;       This routine translates a given value into a an address
                  0D04  ;       based upon the contents of a definition table
                  0D04  ;
                  0D04  ; Inputs:
                  0D04  ;
                  0D04  ;       r2      = value to translate
                  0D04  ;       r3      = address of the definition table
                  0D04  ;
                  0D04  ;       The bit definition table is formatted as follows:
                  0D04  ;
                  0D04  ;       value1,address1
                  0D04  ;       value2,address2
                  0D04  ;       value3,address3
                  0D04  ;       etc.
                  0D04  ;       -1,-1
                  0D04  ;
                  0D04  ;       where "value" is a possible value in r2 and "address" is
                  0D04  ;       the address to be returned when value is encountered
                  0D04  ;
                  0D04  ; Outputs:
                  0D04  ;
                  0D04  ;       condition codes:
                  0D04  ;               NEQ ==> match for value found in the table
                  0D04  ;               EQL ==> no match for value found in the table
                  0D04  ;       r0      = the address associated with value
                  0D04  ;
                  0D04  ; Calling sequence:
                  0D04  ;
                  0D04  ;       movl    test_value, r2          ; setup the test value
                  0D04  ;       movab   table, r3               ; setup table address
                  0D04  ;       jsb     translate_address       ; translate value into address
                  0D04  ;       beql    error                   ; test for no match found
                  0D04  ;
                  0D04  ;---
                  0D04  translate_address::
                  0D04
   50    53   D0  0D04          movl    r3, r0                  ; start the search
   52    80   D1  0D07  10$:    cmpl    (r0)+, r2               ; does value match table entry?
         07   13  0D0A          beql    50$                     ; branch if no match
         80   D5  0D0C          tstl    (r0)+                   ; move to next value in table
         F7   18  0D0E          bgeq    10$                     ; branch if next entry exists
         50   D4  0D10          clrl    r0                      ; else signal no match
              05  0D12          rsb                             ; and exit
                  0D13
                  0D13                                          ; found a match
   50    60   D0  0D13  50$:    movl    (r0), r0                ; pickup address entry from table
              05  0D16          rsb                             ; and return
```

```
0D17  2091
0D17  2092          .END
```

```
$$.TMP1                  = 00000001
$$.TMP2                  = 000000EF
ADDRESS                    ********    X   04
ADD_NEW_ENTRY              00000278 R      04
ADD_SYMBOL                 00000215 RG     04
ALLOCATE                   000003D4 RG     04
ALPFL                      00000010 R      03
ALP_TRAVERSE               000006EC R      04
ARGS                     = 00000001
BUFFER                     ********    X   04
BUS_EXT                    0000079F R      02
CLU$GL_CLUB                00000000 RG     02
CLU$GL_LOA_ADDR            00000010 RG     02
CLUSTRLOA                  0000069D R      02
COLUMNS                  = 00000003
COMPARE_ALPHA              000002C7 R      04
COMPARE_VALUE              000002BF R      04
CTL$AL_STACK               00000024 RG     02
CTL$AL_STACKLIM            00000035 RG     02
CTL$GL_CCBBASE             00000049 RG     02
CTL$GL_IMGHDRBF            0000005C RG     02
CTL$GW_CHINDX              00000070 RG     02
DC$_BUS                    ********    X   02
DC$_DISK                   ********    X   02
DC$_JOURNAL                ********    X   02
DC$_MAILBOX                ********    X   02
DC$_TAPE                   ********    X   02
DC$_TERM                   ********    X   02
DEALLOCATE                 000003F2 RG     04
DEV$V_ELG                  ********    X   04
DEV$V_MSCP                 ********    X   04
DEV$V_NET                  ********    X   04
DEV$V_RTT                  ********    X   04
DEVCLASS_TABLE             000006F3 R      02
DEVICE_ROUT                0000072B R      02
DISKS_TAPES                00000813 R      02
DPT$C_LENGTH             = 00000038
DPT$L_FLINK              = 00000000
DPT$T_NAME               = 00000020
DYN$C_SPECIAL            = 00000080
DYN$C_SUBTYPE            = 00000060
DYN_MAINPTR                ********    X   04
DYN_MAP                    ********    X   04
DYN_PTR                    ********    X   04
DYN_SUBPTR                 ********    X   04
DYN_TAB                    ********    X   04
ERRORLOG_EXT               000007AD R      02
EXE$ALLOCBUF               000005AB RG     02
EXE$AL_STACKS              000000FE RG     02
EXE$DEANONPAGED            000005BC RG     02
EXE$GB_CPUTYPE             00000082 RG     02
EXE$GL_INTSTK              00000110 RG     02
EXE$GL_MP                  000000A9 RG     02
EXE$GL_NONPAGED            00000095 RG     02
EXE$GL_PAGED               000000B7 RG     02
EXE$GL_RPB                 000000C8 RG     02
EXE$GL_SPLITADR            000000D7 RG     02
EXE$GQ_SYSTIME             000000EB RG     02
EXE$IPID_TO_EPID           00000667 RG     02
EXE$MCHK                   00000122 RG     02
EXE$MCHK_PRTCT             000005D0 RG     02
FAB$L_STV                  ********    X   04
FORMAT                     000007FB RG     04
FPEMUL                     000006BA R      04
GETMEM                     ********    X   04
GET_SYMBOL                 00000141 RG     04
INSERT                     000002E3 R      04
IOC$GL_DEVLIST             0000012F RG     02
IOC$GL_DPTLIST             00000142 RG     02
IOC$GL_IRPCNT              00000155 RG     02
IOC$GL_IRPFL               00000167 RG     02
IOC$GL_LRPCNT              00000178 RG     02
IOC$GL_LRPFL               0000018A RG     02
IOC$GL_LRPSIZE             0000019B RG     02
IOC$GL_LRPSPLIT            000001AE RG     02
IOC$GL_SRPCNT              000001C2 RG     02
IOC$GL_SRPFL               000001D4 RG     02
IOC$GL_SRPSIZE             000001E5 RG     02
IOC$GL_SRPSPLIT            000001F8 RG     02
IOC$RETURN                 0000020C RG     02
IRP$B_TYPE               = 0000000A
JOURNAL_EXT                0000072F R      02
LCK$GL_HASHTBL             0000021B RG     02
LCK$GL_HTBLCNT             0000022E RG     02
LCK$GL_IDTBL               00000241 RG     02
LCK$GL_MAXID               00000252 RG     02
LDISK_EXT                  0000092B R      02
LENGTH_SYM                 0000012E R      04
LIB$FREE_VM                ********    X   04
LIB$GET_VM                 ********    X   04
LIB$SIGNAL                 ********    X   04
LINE_COUNT                 ********    X   04
LTERM_EXT                  0000075A R      02
L_LEN                    = 00000007
MAILBX_EXT                 0000073E R      02
MCHK                       00000687 R      02
MMG$AL_SYSPCB              00000263 RG     02
MMG$FRSTRONLY              00000275 RG     02
MMG$GL_FPEMUL_BASE         000002F8 RG     02
MMG$GL_GPTE                00000299 RG     02
MMG$GL_MAXGPTE             000002A9 RG     02
MMG$GL_MAXPFN              000002BC RG     02
MMG$GL_NPAGEDYN            0000036D RG     02
MMG$GL_NPAGNEXT            00000381 RG     02
MMG$GL_PAGEDYN             00000395 RG     02
MMG$GL_RMSBASE             000002CE RG     02
MMG$GL_SBR                 00000327 RG     02
MMG$GL_SPTBASE             00000348 RG     02
MMG$GL_SPTLEN              00000336 RG     02
MMG$GL_SYSLOA_BASE         000002E1 RG     02
MMG$GL_SYSPHD              0000035B RG     02
MMG$GL_VAXEMUL_BASE        0000030F RG     02
MMG$IMGHDRBUF              00000287 RG     02
MP                         00000693 R      02
```

```
MSCP                             000006AE R       02
MSCP_EXT                         0000089A R       02
MSG$_BADGSD                      ********     X   04
MSG$_INVBLKTYP                   ********     X   04
MSG$_NOREQ                       ********     X   04
MSG$_NOSYMBOLS                   ********     X   04
MSG$_SUCCESS                     ********     X   04
NETMBX_EXT                       0000074C R       02
NEW_PAGE                         ********     X   04
NODE$C_LENGTH                  = 0000000E
NODE$L_LEFT                    = 00000000
NODE$L_PTR                     = 0000000A
NODE$L_RIGHT                   = 00000004
NODE$W_BAL                     = 00000008
NSYMBOLS                         00000014 R       03
NULL                             00000979 R       04
OBJ$B_GSD_TYPE                 = 00000000
OBJ$B_TYPE                     = 00000000
OBJ$C_GSD                      = 00000001
OBJ$C_GSD_EPM                  = 00000002
OBJ$C_GSD_PRO                  = 00000003
OBJ$C_GSD_PSC                  = 00000000
OBJ$C_GSD_SYM                  = 00000001
OBJ$L_SYM_VALUE                = 00000005
OBJ$T_EPM_NAME                 = 0000000B
OBJ$T_PSC_NAME                 = 00000008
OBJ$T_SYM_NAME                 = 00000009
OBJ$V_SYM_DEF                  = 00000001
OBJ$W_SYM_FLAGS                = 00000002
OFFSET_TABLE                     00000018 R       03
OPTIONS                          ********     X   04
PFN$AB_STATE                     000003BB RG      02
PFN$AB_TYPE                      000003CC RG      02
PFN$AL_BAK                       000003DC RG      02
PFN$AL_HEAD                      00000442 RG      02
PFN$AL_HILIMIT                   000003EB RG      02
PFN$AL_LOLIMIT                   000003FE RG      02
PFN$AL_PTE                       00000411 RG      02
PFN$AW_REFCNT                    00000452 RG      02
PFN$AX_BLINK                     00000420 RG      02
PFN$AX_FLINK                     00000431 RG      02
PFN$AX_WSLX                      00000464 RG      02
PFN$A_BASE                       00000474 RG      02
PHD$L_ESP                      = 0000007C
PHD$L_KSP                      = 00000078
PHD$L_P0BR                     = 000000C8
PHD$L_P0LRASTL                 = 000000CC
PHD$L_P1BR                     = 000000D0
PHD$L_P1LR                     = 000000D4
PHD$L_PC                       = 000000C0
PHD$L_PSL                      = 000000C4
PHD$L_R0                       = 00000088
PHD$L_R1                       = 0000008C
PHD$L_R10                      = 000000B0
PHD$L_R11                      = 000000B4
PHD$L_R12                      = 000000B8
PHD$L_R13                      = 000000BC
PHD$L_R2                       = 00000090
PHD$L_R3                       = 00000094
PHD$L_R4                       = 00000098
PHD$L_R5                       = 0000009C
PHD$L_R6                       = 000000A0
PHD$L_R7                       = 000000A4
PHD$L_R8                       = 000000A8
PHD$L_R9                       = 000000AC
PHD$L_SSP                      = 00000080
PHD$L_USP                      = 00000084
PHV$GL_PIXBAS                    000005E3 RG      02
PIO$GW_IIOIMPA                   000003A8 RG      02
PREFIX_LEN                     = 00000003
PRINT                            ********     X   04
PRINT_LIST                       000005E2 R       04
PRINT_PAGE                       00000696 R       04
PRINT_SYMBOLS                    000005A5 RG      04
PRTBUFLEN                      = 00000024
PRTBUFSIZ                      = 0000002C
PRTCTL1                          000000E8 R       04
PRTCTL2                          000000FB R       04
RAB$W_RSZ                      = 00000022
READ_SYMBOLS                     0000040E RG      04
REG_TABLE                        00000000 R       04
RELOCATE_BASE                    ********     X   04
REQMEM                           ********     X   04
REQ_SYMBOLS                      00000000 R       02
REQ_SYMBOLS_LEN                = 0000067C
REWIND_STB                       00000138 RG      04
RMS                              0000067C R       02
RMS$_EOF                         ********     X   04
ROWS                           = 00000036
RTERM_EXT                        0000077C R       02
SCH$GL_CURPCB                    00000483 RG      02
SCH$GL_FREECNT                   00000495 RG      02
SCH$GL_MAXPIX                    000004BA RG      02
SCH$GL_PCBVEC                    000004A8 RG      02
SCH$GQ_SUSP                      0000061D RG      02
SCH$QAST                         000005F5 RG      02
SCH$WAITK                        0000060F RG      02
SCH$WAKE                         00000602 RG      02
SCS$GA_EXISTS                    000004CC RG      02
SCS$GA_LOCALSB                   000004DE RG      02
SCS$GL_CDL                       000004F1 RG      02
SCS$GL_MSCP                      00000500 RG      02
SCS$GL_PDT                       00000510 RG      02
SCS$GL_RDT                       0000051F RG      02
SCS$GQ_CONFIG                    0000052E RG      02
SCS$GQ_DIRECT                    00000540 RG      02
SCSLOA                           000006E5 R       02
SET_HEADING                      ********     X   04
SGN$GL_BALSETCT                  0000062D RG      02
SGN$GL_PAGEDYN                   00000552 RG      02
SGN$GW_ISPPGCT                   00000565 RG      02
SORT_OFFSETS                     00000BDE R       04
STB                              ********     X   04
STBF                             ********     X   04
```

```
STB_BUFFER                 ********    X     04
STB_PTR                    00000000 R        03
STORAGE_SIZE             = 00001BD8
STRUCTURE                  00000020 RG       03
SWP$A_KSTK                 00000578 RG       02
SWP$GL_BALBASE             00000654 RG       02
SWP$GL_BSLOTSZ             00000641 RG       02
SWP$K_KSTKSZ               00000587 RG       02
SYM$C_LENGTH             = 00000024
SYM$L_VALUE              = 00000000
SYM$T_SYMBOL             = 00000004
SYMBOL_SIZE                0000072F RG       04
SYMBOL_VALUE               000007BA RG       04
SYMFL                      0000000C R        03
SYS$CLOSE                  ********   GX     04
SYS$FAO                    ********    X     04
SYS$GET                    ********   GX     04
SYS$GQ_VERSION             00000598 RG       02
SYSLOA                     000006D7 R        02
TABLE_INSERT               00000C66 R        04
TPA$L_NUMBER             = 0000001C
TRANSCTL1                  00000113 R        04
TRANSCTL2                  00000123 R        04
TRANSLATE_ADDRESS          00000D04 RG       04
TRANSLATE_BITS             00000CC0 RG       04
TRANSLTH                   00000004 R        03
TRAVERSE                   000006C7 R        04
TRYMEM                     ********    X     04
TYPELEN                  = 00000008
TYPETAB                    000007F3 R        04
UCB$B_DEVCLASS           = 00000040
UCB$C_LENGTH             = 00000090
UCB$L_DEVCHAR            = 00000038
UCB$L_DEVCHAR2           = 0000003C
UCB$W_SIZE               = 00000008
UCB_BUS                    00000B67 R        04
UCB_DT                     00000B0D R        04
UCB_JOURNAL                00000BA8 R        04
UCB_MAIL                   00000B79 R        04
UCB_PREFIX                 00000135 R        04
UCB_SORT_OFFSETS           00000A35 R        04
UCB_STRUCT                 00000028 R        03
UCB_TERM                   00000B44 R        04
VALID_UCBSYMBOL            00000BBA R        04
VAXEMUL                    000006C8 R        02
```

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

PSECT name                 Allocation          PSECT No.  Attributes
----------                 ----------          ---------  ----------
.  ABS  .                  00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                      00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
ZREQSYMS                   000009D3  ( 2515.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC BYTE
SDADATA                    000000B8  (  184.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC BYTE
SYMBOLS                    00000D17  ( 3351.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC BYTE
```

```
LITERALS                000001CA  (  458.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC BYTE

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

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                     29     00:00:00.04    00:00:00.93
Command processing                107     00:00:00.46    00:00:03.37
Pass 1                            492     00:00:12.77    00:00:48.93
Symbol table sort                   0     00:00:01.62    00:00:05.21
Pass 2                            393     00:00:04.26    00:00:13.09
Symbol table output                32     00:00:00.17    00:00:00.61
Psect synopsis output               3     00:00:00.02    00:00:00.02
Cross-reference output              0     00:00:00.00    00:00:00.00
Assembler run totals             1058     00:00:19.34    00:01:12.16

The working set limit was 2400 pages.
122713 bytes (240 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1425 non-local and 168 local symbols.
2092 source lines were read in Pass 1, producing 63 object records in Pass 2.
42 pages of virtual memory were used to define 40 macros.

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

Macro library name                               Macros defined
------------------                               --------------
_$255$DUA28:[SDA.OBJ]SDALIB.MLB;1                      13
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         10
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      11
TOTALS (all libraries)                                 34

1506 GETS were required to define 34 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYMBOLS/OBJ=OBJ$:SYMBOLS MSRC$:SYMBOLS/UPDATE=(ENH$:SYMBOLS)+EXECML$/LIB+LIB$:SDALIB/LIB
```

0354 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYMBOLS
LIS

SMGRTL

SMGBLDTRM
MAP

SDAMSG
LIS

VAXINST
LIS

SMGMAPTRM
MAP

SMGKCB
SDL

VALIDATE
LIS

STACKS
LIS

SMGDEF
SDL

SMGKDE
SDL

SMGSHR
MAP