I 3

```
   CCCCCCCCCCCC  LLL                 IIIIIIIIII     UUU         UUU TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCCC  LLL                 IIIIIIIIII     UUU         UUU TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCCC  LLL                 IIIIIIIIII     UUU         UUU TTTTTTTTTTTTTTT  LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
CCC              LLL                    III         UUU         UUU       TTT        LLL
   CCCCCCCCCCCC  LLLLLLLLLLLLLL      IIIIIIIIII     UUUUUUUUUUUUUUU       TTT        LLLLLLLLLLLLLL
   CCCCCCCCCCCC  LLLLLLLLLLLLLL      IIIIIIIIII     UUUUUUUUUUUUUUU       TTT        LLLLLLLLLLLLLL
   CCCCCCCCCCCC  LLLLLLLLLLLLLL      IIIIIIIIII     UUUUUUUUUUUUUUU       TTT        LLLLLLLLLLLLLL
```

FILEID**SHWCLSTR

```
  SSSSSSSS  HH      HH  WW      WW    CCCCCCCC  LL            SSSSSSSS  TTTTTTTTTT  RRRRRRRR
  SSSSSSSS  HH      HH  WW      WW    CCCCCCCC  LL            SSSSSSSS  TTTTTTTTTT  RRRRRRRR
SS          HH      HH  WW      WW  CC          LL          SS              TT      RR      RR
SS          HH      HH  WW      WW  CC          LL          SS              TT      RR      RR
SS          HH      HH  WW      WW  CC          LL          SS              TT      RR      RR
SS          HH      HH  WW      WW  CC          LL          SS              TT      RR      RR
  SSSSSS    HHHHHHHHHH  WW      WW  CC          LL            SSSSSS        TT      RRRRRRRR
  SSSSSS    HHHHHHHHHH  WW      WW  CC          LL            SSSSSS        TT      RRRRRRRR
        SS  HH      HH  WW  WW  WW  CC          LL                  SS      TT      RR  RR
        SS  HH      HH  WW  WW  WW  CC          LL                  SS      TT      RR  RR
        SS  HH      HH  WWWW  WWWW  CC          LL                  SS      TT      RR    RR        ....
        SS  HH      HH  WWWW  WWWW  CC          LL                  SS      TT      RR    RR        ....
SSSSSSSS    HH      HH  WW      WW    CCCCCCCC  LLLLLLLLLL  SSSSSSSS        TT      RR      RR      ....
SSSSSSSS    HH      HH  WW      WW    CCCCCCCC  LLLLLLLLLL  SSSSSSSS        TT      RR      RR      ....

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

```
0000        .SBTTL  Program description
0000  ;++
0000  ;
0000  ; FACILITY:
0000  ;
0000  ;       VAX/VMS SHOW CLUSTER utility
0000  ;
0000  ;
0000  ; ABSTRACT:
0000  ;
0000  ;       SHWCLSTR is a utility used to display infomation about CI clusters and
0000  ;       the local hardware interface to the cluster.  Contained within are the
0000  ;       routines necessary to extract this information from the SCS database
0000  ;       and the CI port database and present the information to the user in a
0000  ;       readable format.  Furthermore, all routines needed to parse and process
0000  ;       input from the user are also included here.
0000  ;
0000  ;
0000  ; AUTHOR:        Paul R. DeStefano,       July, 1983
0000  ;
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ;       V03-016 PRD0111         Paul R. DeStefano       31-Jul-1984
0000  ;               Update FIRST_DATA_LINE, which contains the number of
0000  ;               first line of the scrolling region, when the CLUSTER
0000  ;               class is added or removed from the display because
0000  ;               the ADD or REMOVE field command was entered and the
0000  ;               field was the first CLUSTER class field being added
0000  ;               or the last being removed.
0000  ;
0000  ;       V03-015 PRD0107         Paul R. DeStefano       20-Jul-1984
0000  ;               Correct formating of descriptor passed to $GETDVI.
0000  ;               Make INIT assume that output goes to a listing file
0000  ;               if $GETDVI fails.
0000  ;               Make sure initialization file gets executed when
0000  ;               running in a Batch job.
0000  ;               Change scrolling back to original design.
0000  ;
0000  ;       V03-014 PRD0103         Paul R. DeStefano       11-Jul-1984
0000  ;               Make sure DISPLAY_COMMON assumes successful return.
0000  ;               Fix access violation in OUTPUT_TIME routine.
0000  ;
0000  ;       V03-013 PRD0100         Paul R. DeStefano       08-May-1984
0000  ;               Modified RESET routine to properly restore the init
0000  ;               flag.
0000  ;
0000  ;       V03-012 PRD0098         Paul R. DeStefano       30-Apr-1984
0000  ;               Fix DO_SNAPSHOT routine so that variable length buffers
0000  ;               are unlocked.
0000  ;
0000  ;       V03-011 PRD0095         Paul R. DeStefano       11-Apr-1984
0000  ;               Removed COORD_STATUS and LOCAL_STATUS fields.
0000  ;
0000  ;       V03-010 PRD0089         Paul R. DeStefano       29-Mar-1984
0000  ;               Fix display of CLUSTER class title when the ADD CLUSTER
0000  ;               command is used in an initialization file.
```

```
0000   ;
0000   ;       V03-009 PRD0080         Paul R. DeStefano       25-Mar-1984
0000   ;               Store only the prefix of offset name to save space in
0000   ;               field descriptors.
0000   ;               Added code to format a 15 character time string to
0000   ;               FORMAT_DATA routines.
0000   ;               Improved error messages.
0000   ;               Improve output of time.
0000   ;               Improve method of limiting up- and left-scrolling.
0000   ;               Add SECOND_CHANCE routine.
0000   ;               Limit refreshing on commands that fail.
0000   ;               Added TRANS_TIME and QD_NAME fields.
0000   ;               Removed FORM_STATUS field.
0000   ;
0000   ;       V03-008 PRD0079         Paul R. DeStefano       11-Mar-1984
0000   ;               Don't do refresh after SET INTERVAL or SET SCREEN commands.
0000   ;               Correct output when using SYS$OUTPUT is redefined using
0000   ;               DEFINE/USER.  Correct output when I/O is to a mailbox.
0000   ;               Correct output when /OUTPUT is used within a BATCH job.
0000   ;               Correct maximum data width when ASCIC format is used with
0000   ;               a display routine other than DISPLAY_FIELD.
0000   ;
0000   ;       V03-007 PRD0076         Paul R. DeStefano       28-Feb-1984
0000   ;               Correct output file name when SYS$OUTPUT is defined as a
0000   ;               file.
0000   ;               Add new fields to CLUSTER and MEMBERS classes.
0000   ;               Make sure Local System Block is displayed when VMSVAXCLUSTER
0000   ;               doesn't equal 1.
0000   ;               Make sure line drawing and special graphics character are set
0000   ;               in G1 area in DECcrt on each screen update.
0000   ;               Call SET_REGION_FULL before exiting for foreign terminals
0000   ;               emulating VT100s.
0000   ;               Perform initial syntax checking using DCL_PARSE.
0000   ;               Improved parsing of lists of parameters and qualifier values.
0000   ;               Added ADD_ALL_FIELDS routine for new ADD class/ALL qualifier.
0000   ;               Make INTERVAL value = seconds rather than delta time.
0000   ;               Improved message handling.
0000   ;               Modified REMOVE_FIELD routine to turn off a class when the
0000   ;               last field is removed.
0000   ;               Display CLUSTER class at the top of the CLUSTER report.
0000   ;
0000   ;       V03-006 PRD0046         Paul R. Destefano       11-Jan-1984
0000   ;               Correct the number of bytes displayed by the SYS_ID field.
0000   ;               Make COUNTERS a separate class.
0000   ;               Retry snapshot with expanded buffers if snapshot buffers
0000   ;               overflow.
0000   ;
0000   ;       V03-005 PRD0043         Paul R. DeStefano       27-Sep-1983
0000   ;               Initialize connection count in path blocks at
0000   ;               PB$SHC$L_NUM_CDT so that a circuit losing all
0000   ;               connections will be properly detected.
0000   ;               Fix access violation in CHECK_POS routine.
0000   ;
0000   ;       V03-004 PRD0042         Paul R. DeStefano       27-Sep-1983
0000   ;               Fix setup of initialization file default so that an
0000   ;               extention of .INI is defaulted but not forced.
0000   ;               Add additional remote port types to RP_TYPE field.
```

```
0000  ;
0000  ;       V03-003 PRD0041         Paul R. DeStefano       15-Sep-1983
0000  ;               Modified default report display.
0000  ;               Fix handling of formative circuits in CHECK_POS routine.
0000  ;               Add variable buffer address (VAR_BUF_ADRS) to locked
0000  ;               portion of working set.
0000  ;               Fix refresh after INITIALIZE command.
0000  ;
0000  ;       V03-002 PRD0028         Paul R. DeStefano       23-Aug-1983
0000  ;               Modified handling of broadcast messages to allow for
0000  ;               messages that lack carriage control characteristics.
0000  ;
0000  ;               Fixed handling of an initialization command file that
0000  ;               contains an INITIALIZE command.
0000  ;
0000  ;--
0000
```

```
0000
0000            .SBTTL  Definitions.
0000    ;
0000    ; System definitions: (From SYS$LIBRARY:LIB.MLB)
0000    ;
0000
0000            $CDLDEF                                 ; Define connection descriptor list
0000                                                    ;  offsets.
0000            $CDTDEF                                 ; Define connection descriptor table
0000                                                    ;  offsets.
0000            $CIBDDEF                                ; Define buffer descriptor offsets.
0000            $CIBDTDEF                               ; Define buffer descriptor table
0000                                                    ;  offsets.
0000            $CLUBDEF                                ; Define Cluster Block.
0000            $CSBDEF                                 ; Define Cluster System Block.
0000            $DCDEF                                  ; Define device class codes.
0000            $DDBDEF                                 ; Define DDB offsets.
0000            $DIBDEF                                 ; Define device characteristics offsets.
0000            $DVIDEF                                 ; Define item identifier codes.
0000            $HLPDEF                                 ; Define help flags.
0000            $IPLDEF                                 ; Define processor priority levels.
0000            $MSGDEF                                 ; Define message types.
0000            $PBDEF                                  ; Define path block offsets.
0000            $PDTDEF                                 ; Define port descriptor table offsets.
0000            $PRVDEF                                 ; Define privilege mask bits.
0000            $SBDEF                                  ; Define system block offsets.
0000            $SCRDEF                                 ; Define screen package parameters.
0000            $SCSDEF                                 ; Define SCS message formats.
0000            $SDIRDEF                                ; Define SCS directory entry offsets.
0000            $SSDEF                                  ; Define status codes.
0000            $TPADEF                                 ; Define LIB$TPARSE parameter block.
0000            $TRMDEF                                 ; Define terminal function modifiers.
0000            $TTDEF                                  ; Define terminal characteristics.
0000            $TT2DEF                                 ; Define extended terminal
0000                                                    ;  characteristics.
0000            $UCBDEF                                 ; Define UCB offsets.
0000
0000    ;
0000    ; CI specific definitions: (From PALIB.MLB)
0000    ;
0000
0000            $PAPDTDEF                               ; Define the extention to the port
0000                                                    ;  descriptor table.
0000            $PPDDEF                                 ; Define datagram offsets.
0000
0000    ;
0000    ; Define offsets into field descriptors.
0000    ;
0000
0000            $DEFINI FD
0000
0000            $DEF    FD$B_FIELD_SIZE         .BYTE   ; Number of columns currently
0001                                                    ;  occupied by the field.
0001            $DEF    FD$B_MIN_FLD_SIZE       .BYTE   ; Minimum number of columns that
0002                                                    ;  can be used to display the
0002                                                    ;  field.
0002            $DEF    FD$B_MAX_DATA_SIZE      .BYTE   ; Maximum number of columns
```

```
                 0003                                                       ;  needed to display the data.
                 0003          $DEF     FD$B_MAX_FLD_SIZE      .BYTE        ; Maximum number of columns
                 0004                                                       ;  needed to display the field.
                 0004                                                       ;  (For both title and data.)
                 0004          $DEF     FD$B_LENGTH            .BYTE        ; Number of bytes of data to
                 0005                                                       ;  extract from the SCS
                 0005                                                       ;  database to display the
                 0005                                                       ;  field.
                 0005          $DEF     FD$B_FORMAT            .BYTE        ; Type of format to use when
                 0006                                                       ;  displaying the field.
                 0006                                                       ;
        00000008 0006     .=.+2                                             ; Reserve one word.
                 0008
                 0008          $DEF     FD$L_DISPLAY_ROUTINE   .LONG        ; Routine to call to display the
                 000C                                                       ;  field.
                 000C          $DEF     FD$L_CLASS             .LONG        ; Class to which the field
                 0010                                                       ;  belongs.
                 0010          $DEF     FD$L_OFFSET            .LONG        ; Value of the SCS offset.
                 0014          $DEF     FD$L_OFFSET_PRFX       .LONG        ; First four characters of
                 0018                                                       ;  offset name string.
                 0018          $DEF     FD$B_TITLE_SIZE        .BYTE        ; Number of characters in the
                 0019                                                       ;  title for the field.
                 0019          $DEF     FD$T_TITLE                          ; Offset to the title string.
                 0019
                 0019          $DEFEND FD,,DEF
                 0000
                 0000
                 0000     ;
                 0000     ; Define the field display format types.
                 0000     ;
                 0000
        00000001 0000     ASCII           =        1                  ; Data is to be formatted in ascii.
        00000002 0000     HEX             =        2                  ; Data is to be formatted in hexadecimal
        00000003 0000     DEC             =        3                  ; Data is to be formatted in decimal.
        00000004 0000     ASCIC           =        4                  ; Data is to be formatted using a
                 0000                                                 ;  counted ascii string.
        00000005 0000     TIME            =        5                  ; Data is to be formatted as a time
                 0000                                                 ;  string.
        0000000F 0000     TIME_STRING_LEN =        15                 ; Length of formatted time string.
                 0000
                 0000     ;
                 0000     ; Define lengths and starting offsets of special buffers for snapshot data.
                 0000     ;
                 0000
                 0000     ;
                 0000     ; The portion of the CDT stored starts at CDT$L_RCONID.  The length is the
                 0000     ; number of bytes in CDT$L_RCONID through CDT$W_QBDT_CNT plus the forward link.
                 0000     ; The difference is a negative value to be added to the SCS offset in the
                 0000     ; field descriptors to compensate for the portion of the CDT that isn't stored.
                 0000     ;
                 0000
        00000014 0000     SHC$L_CDT_BEGIN =       CDT$L_RCONID                          ; Starting offset.
        0000008C 0000     SHC$C_CDT_LENGTH=       <<CDT$W_QBDT_CNT+2>-CDT$L_RCONID>+4     ; Length.
        FFFFFFF0 0000     SHC$L_CDT_DIF   =       <CDT$L_MSGINPUT-CDT$L_RCONID>+4 ; Difference.
                 0000
                 0000     ;
                 0000     ; The portion of the PDT stored starts at PDT$L_MSGHDRSZ.  The length is the
```

```
          0000  ; number of bytes in PDT$L_MSGHDRSZ through PDT$L_LBDG.  The difference is a
          0000  ; negative value to be added to the SCS offset in the field descriptors to
          0000  ; compensate for the portion of the PDT that isn't stored.
          0000
000000B4  0000  SHC$L_PDT_BEGIN =       PDT$L_MSGHDRSZ                  ; Starting offset.
000000D0  0000  SHC$L_PDT_LENGTH=       PDT$L_LBDG-PDT$L_MSGHDRSZ       ; Length.
FFFFFF4C  0000  SHC$L_PDT_DIF   =       <PDT$W_SIZE-8>-PDT$L_MSGHDRSZ   ; Difference.
          0000
          0000
          0000  ;
          0000  ; Define offsets for storing the number of circuits on a system and the number
          0000  ; of connections on a circuit.
          0000  ;
          0000
00000004  0000  SB$SHC$L_NUM_PB =       SB$L_BLINK      ; Store the number of paths block on a
          0000                                          ;   system block where the back link
          0000                                          ;   would normally be.
00000004  0000  PB$SHC$L_NUM_CDT=       PB$L_BLINK      ; Store the number of connections on a
          0000                                          ;   path block where the back link would
          0000                                          ;   normally be.
          0000  ;
          0000  ; Define the offset for storing the display's line number in the system block.
          0000  ; This is used to associate a system with a line in the display to insure that
          0000  ; a line which has moved in the display, but has not changed does not go
          0000  ; undetected.
          0000  ;
          0000
00000009  0000  SHC$SB$B_DIS_POS=       SB$W_SIZE+1     ; Store in upper byte of size field in
          0000                                          ;   system block.
          0000
          0000  ;
          0000  ; Define offsets into MISC_BUF.
          0000  ;
          0000
          0000          $DEFINI MB
          0000
          0000          $DEF    MB$L_PORT_NAME  .LONG   ; Port name.
          0004          $DEF    MB$L_PORT_NUM   .LONG   ; Port number.
          0008          $DEF    MB$L_DEVTYPE    .LONG   ; Device type.
          000C          $DEF    MB$L_STS        .LONG   ; Device status.
          0010          $DEF    MB$B_ERTCNT     .BYTE   ; # of reinitialize attempts remaining.
          0011          $DEF    MB$B_ERTMAX     .BYTE   ; Maximum number of reinits allowed.
          0012          $DEF    MB$W_ERRCNT     .WORD   ; Number of errors logged.
          0014          $DEF    MB$L_FORM_PB    .LONG   ; Number of formative path blocks.
          0018          $DEF    MB$L_PL_WAITERS .LONG   ; Number of SCS non-paged pool waiters.
          001C          $DEF    MB$W_DG_FREE    .WORD   ; Number of free datagram buffers
          001E                                          ;   currently queued for receives.
          001E          $DEF    MB$W_MSG_FREE   .WORD   ; Number of free message buffers
          0020                                          ;   currently queued for receives.
          0020          $DEF    MB$W_CMDS_QUED  .WORD   ; Total number of messages, datagrams,
          0022                                          ;   and port commands queued for
          0022                                          ;   transmission at all priorities.
          0022                                          ;   by this port.
          0022          $DEF    MB$W_RSPS_QUED  .WORD   ; Number of responses of all kinds
          0024                                          ;   received on this port, but not yet
          0024                                          ;   processed.
          0024          $DEFEND MB,,DEF
```

```
          0000
0000002E  0000   MISC_BUF_LEN    =       46              ; 11 longwords in miscellaneous buffer.
          0000
          0000
          0000   ;
          0000   ; Define masks and bit fields for device types.
          0000   ;
          0000
00000000  0000   DVT$V_VT52      =       0               ; VT52 compatable terminal.
00000001  0000   DVT$V_DCRT      =       1               ; VT100 compatable terminal.
00000002  0000   DVT$V_HARD      =       2               ; Hardcopy device.
00000003  0000   DVT$V_UNKNOWN   =       3               ; Unknown type of scope.
00000004  0000   DVT$V_AVO       =       4               ; Advanced video option present.
00000005  0000   DVT$V_DISK      =       5               ; Disk.
          0000
00000001  0000   DVT$M_VT52      =       1@DVT$V_VT52
00000002  0000   DVT$M_DCRT      =       1@DVT$V_DCRT
00000004  0000   DVT$M_HARD      =       1@DVT$V_HARD
00000008  0000   DVT$M_UNKNOWN   =       1@DVT$V_UNKNOWN
00000010  0000   DVT$M_AVO       =       1@DVT$V_AVO
00000020  0000   DVT$M_DISK      =       1@DVT$V_DISK
          0000
          0000   ;
          0000   ; Define the terminal screen widths.
          0000   ;
          0000
00000050  0000   NARROW          =       80              ; DECcrt 80 column mode.
00000084  0000   WIDE            =       132             ; DECcrt 132 column mode.
          0000
          0000
          0000   ;
          0000   ; Define special keyboard characters.
          0000   ;
          0000
0000001B  0000   ESC             =       ^X1B            ; Ascii escape character.
0000009B  0000   CSI             =       ^X9B            ; Eight bit equivalent of "ESC [".
00000017  0000   CTRLW           =       ^X17            ; Ascii control-W.
0000001A  0000   CTRLZ           =       ^X1A            ; Ascii control-Z.
          0000
          0000
          0000   ;
          0000   ; Define the escape sequences sent by the arrow keys.
          0000   ;
          0000
0000415B  0000   DCRT_UP_ARROW   =       ^A/[A/          ; DECcrt up arrow.
0000425B  0000   DCRT_DWN_ARROW  =       ^A/[B/          ; DECcrt down arrow.
0000435B  0000   DCRT_LFT_ARROW  =       ^A/[C/          ; DECcrt left arrow.
0000445B  0000   DCRT_RT_ARROW   =       ^A/[D/          ; DECcrt right arrow.
          0000
00000041  0000   VT52_UP_ARROW   =       ^A/A/           ; VT52 up arrow.
00000042  0000   VT52_DWN_ARROW  =       ^A/B/           ; VT52 down arrow.
00000043  0000   VT52_LFT_ARROW  =       ^A/C/           ; VT52 left arrow.
00000044  0000   VT52_RT_ARROW   =       ^A/D/           ; VT52 right arrow.
          0000
          0000
          0000   ;
          0000   ; Special characters to turn the special graphics characters on and off.
```

```
          0000  ;
          0000
0000000E  0000  GRAPHICS_ON    =    ^X0E              ; Control_N or SO (shift-out)
          0000
0000000F  0000  GRAPHICS_OFF   =    ^X0F              ; Control_O or SI (shift-in)
          0000
          0000
          0000  ;
          0000  ; DECcrt special graphics characters
          0000  ;
          0000
0000006A  0000  LR_CORNER      =    ^A/j/             ; Lower-right corner.
0000006B  0000  UR_CORNER      =    ^A/k/             ; Upper-right corner.
0000006C  0000  UL_CORNER      =    ^A/l/             ; Upper-left corner.
0000006D  0000  LL_CORNER      =    ^A/m/             ; Lower-left corner.
0000006E  0000  CROSS          =    ^A/n/             ; Crossing lines.
00000071  0000  SCAN_5         =    ^A/q/             ; Horizontal line - Scan 5.
00000074  0000  LEFT_T         =    ^A/t/             ; Left "T".
00000075  0000  RIGHT_T        =    ^A/u/             ; Right "T".
00000076  0000  BOTTOM_T       =    ^A/v/             ; Bottom "T".
00000077  0000  TOP_T          =    ^A/w/             ; TOP "T".
00000078  0000  VERT_BAR       =    ^A/x/             ; Vertical bar.
          0000
          0000
          0000  ;
          0000  ; Define masks and bit fields for the type of information currently output
          0000  ; on the two command input lines.
          0000  ;
          0000
00000000  0000  CL$V_BRDCST    =    0                 ; Broadcast message.
00000001  0000  CL$V_CMD       =    1                 ; Command input.
00000002  0000  CL$V_ERROR     =    2                 ; Error message.
00000003  0000  CL$V_SCROLL    =    3                 ; Scroll message.
          0000
00000001  0000  CL$M_BRDCST    =    1@CL$V_BRDCST
00000002  0000  CL$M_CMD       =    1@CL$V_CMD
00000004  0000  CL$M_ERROR     =    1@CL$V_ERROR
00000008  0000  CL$M_SCROLL    =    1@CL$V_SCROLL
          0000
          0000
          0000
          0000  ;
          0000  ; Define commonly used line numbers.
          0000  ;
          0000
00000002  0000  HEADER_START   =    2                 ; Display header starts at line two.
00000007  0000  HEADER_END     =    HEADER_START+5    ;  and is five lines long.
          0000
          0000
          0000  ;
          0000  ; Define the mask for CURRENT_REPORT.
          0000  ;
          0000
00000000  0000  RPT$V_CLSTR    =    0                 ; Set for cluster report, clear for
00000001  0000  RPT$M_CLSTR    =    1@RPT$V_CLSTR     ;  local report.
          0000
          0000
```

```
                  0000      ;
                  0000      ; Define ASCIC strings used to identify device descriptor blocks.
                  0000      ;
                  0000
        41415003  0000      PAAC            =       ^X41415003      ; ''PAA''.
        42415003  0000      PABC            =       ^X42415003      ; ''PAB''
                  0000
                  0000
                  0000      ;
                  0000      ; To add a new class.
                  0000      ;
                  0000      ;       1)      Define the class mask and field bits.
                  0000      ;       2)      Add the ascii class name @CLASS_NAMES
                  0000      ;       3)      Add a longword to the TITLE_LEN_ARRAY
                  0000      ;       4)      Add field definitions preceeding them with a
                  0000      ;               CLASS = class_mask statement.
                  0000      ;       5)      Increase NUM_CLASSES by 1.
                  0000      ;
                  0000
                  0000      ;
                  0000      ; To add a new data structure for display by DISPLAY_FIELD.
                  0000      ;
                  0000      ;       1)      Add the new prefix to SYM_PRFX.
                  0000      ;       2)      Add a buffer for the new structure (fixed length) or
                  0000      ;               descriptors for a variable buffer.
                  0000      ;       3)      Modify the SNAPSHOT routine to capture the new structure.
                  0000      ;       4)      Add a new GET_DATA call frame index.
                  0000      ;
                  0000
        00000009  0000      NUM_CLASSES     =       9               ; Nine basic class types.
                  0000
                  0000      ;
                  0000      ; Masks for controlling display of classes.
                  0000      ;
                  0000
        00000000  0000      CLS$V_CLUB      =       0               ; Cluster class.
        00000001  0000      CLS$V_SYS       =       1               ; Systems class.
        00000002  0000      CLS$V_CSB       =       2               ; Members class.
        00000003  0000      CLS$V_CIR       =       3               ; Circuits class.
        00000004  0000      CLS$V_CON       =       4               ; Connections class.
        00000005  0000      CLS$V_CNT       =       5               ; Counters class.
        00000006  0000      CLS$V_CRD       =       6               ; Credits class.
        00000007  0000      CLS$V_LPRT      =       7               ; Local ports class.
        00000008  0000      CLS$V_ERR       =       8               ; Errors class.
        00000009  0000      CLS$V_OPEN_CIR  =       9               ; Circuits in the open state.
        0000000A  0000      CLS$V_NOOPEN_CIR=       10              ; Circuits in states other than open.
        0000000B  0000      CLS$V_OPEN_CON  =       11              ; Connections in the open state.
        0000000C  0000      CLS$V_NOOPEN_CON=       12              ; Connections in states other than open.
                  0000
                  0000      ;
                  0000      ; Types of systems.  Must match HW_TYPE_TABLE.
                  0000      ;
                  0000
        00000010  0000      CLS$V_750_SYS   =       16              ; 750 systems.
        00000011  0000      CLS$V_780_SYS   =       17              ; 780 systems.
        00000012  0000      CLS$V_785_SYS   =       18              ; 785 systems.
        00000013  0000      CLS$V_790_SYS   =       19              ; 790 systems.
```

```
00000014  0000   CLS$V_HSC50_SYS =     20              ; HSC50 systems.
00000015  0000   CLS$V_HSC60_SYS =     21              ; HSC60 systems.
          0000
00000001  0000   CLS$M_CLUB      =     1@CLS$V_CLUB
00000002  0000   CLS$M_SYS       =     1@CLS$V_SYS
00000004  0000   CLS$M_CSB       =     1@CLS$V_CSB
00000008  0000   CLS$M_CIR       =     1@CLS$V_CIR
00000010  0000   CLS$M_CON       =     1@CLS$V_CON
00000020  0000   CLS$M_CNT       =     1@CLS$V_CNT
00000040  0000   CLS$M_CRD       =     1@CLS$V_CRD
00000080  0000   CLS$M_LPRT      =     1@CLS$V_LPRT
00000100  0000   CLS$M_ERR       =     1@CLS$V_ERR
          0000
          0000   ;
          0000   ; Mask for any type of SCS class. (Includes MEMBERS)
          0000   ;
          0000
          0000   CLS$M_ANY_SCS   =     <CLS$M_SYS!-    ; SYSTEMS
          0000                          CLS$M_CIR!-    ; CIRCUITS
          0000                          CLS$M_CON!-    ; CONNECTIONS
          0000                          CLS$M_CNT!-    ; COUNTERS
          0000                          CLS$M_CRD!-    ; CREDITS
0000007E  0000                          CLS$M_CSB>     ; MEMBERS
          0000
          0000   ;
          0000   ; Mask for any type of CLUSTER report class.
          0000   ;
          0000
          0000   CLS$M_ANY_CLUS  =     <CLS$M_ANY_SCS!-        ; Any SCS class.
0000007F  0000                          CLS$M_CLUB>            ; Plus the CLUSTER class.
          0000
          0000   ;
          0000   ; Mask for any types of system
          0000   ;
          0000
00020000  0000   CLS$M_780_SYS   =     1@CLS$V_780_SYS         ; 11/780
00040000  0000   CLS$M_785_SYS   =     1@CLS$V_785_SYS         ; 11/785
00010000  0000   CLS$M_750_SYS   =     1@CLS$V_750_SYS         ; 11/750
00100000  0000   CLS$M_HSC50_SYS =     1@CLS$V_HSC50_SYS       ; HSC50
00200000  0000   CLS$M_HSC60_SYS =     1@CLS$V_HSC60_SYS       ; HSC60
          0000
          0000
          0000   CLS$M_ALL_SYS   =     <CLS$M_SYS!-            ; Any type of system...
          0000                          CLS$M_780_SYS!-        ; ...11/780
          0000                          CLS$M_785_SYS!-        ; ...11/785
          0000                          CLS$M_750_SYS!-        ; ...11/750
          0000                          CLS$M_HSC50_SYS!-      ; ...HSC50
00370002  0000                          CLS$M_HSC60_SYS>       ; ...HSC60
          0000
          0000
          0000   ;
          0000   ; Mask for all types of circuits.
          0000   ;
          0000
00000200  0000   CLS$M_OPEN_CIR  =     1@CLS$V_OPEN_CIR        ; Open CIRCUITS.
00000400  0000   CLS$M_NOOPEN_CIR=     1@CLS$V_NOOPEN_CIR      ; Non-open CIRCUITS.
          0000
```

```
                0000  CLS$M_ALL_CIR   =       <CLS$M_CIR!-              ; All types of circuits...
                0000                            CLS$M_OPEN_CIR!-        ; ...Open CIRCUITS
      00000608  0000                            CLS$M_NOOPEN_CIR>       ; ...Non-open CIRCUITS
                0000
                0000  ;
                0000  ; Masks for all types of connections.
                0000  ;
                0000
      00000800  0000  CLS$M_OPEN_CON  =       1@CLS$V_OPEN_CON          ; Open CONNECTIONS
      00001000  0000  CLS$M_NOOPEN_CON=       1@CLS$V_NOOPEN_CON        ; Non-open CONNECTIONS
                0000
                0000  CLS$M_ALL_CON   =       <CLS$M_CON!-              ; All types of CONNECTIONS...
                0000                            CLS$M_OPEN_CON!-        ; ...Open CONNECTIONS
      00001810  0000                            CLS$M_NOOPEN_CON>       ; ...Non-open CONNECTIONS
                0000
                0000  ;
                0000  ; Define the longword indices into the GET_DATA call frame.
                0000  ;
                0000
      00000001  0000  CLUB_IN         =       1
      00000002  0000  SB_IN           =       2
      00000003  0000  CSB_IN          =       3
      00000004  0000  PB_IN           =       4
      00000005  0000  CDT_IN          =       5
      00000006  0000  PDT_IN          =       6
      00000007  0000  MB_IN           =       7
                0000
                0000  ;
                0000  ; Masks for event flags.
                0000  ;
                0000
      00000001  0000  EVF$V_TIM_FLAG  =       1                 ; Refresh timer expired.
      00000002  0000  EVF$V_AST_FLAG  =       2                 ; AST delivery.
                0000
      00000002  0000  EVF$M_TIM_FLAG  =       1@EVF$V_TIM_FLAG
      00000004  0000  EVF$M_AST_FLAG  =       1@EVF$V_AST_FLAG
                0000
                0000  ;
                0000  ; Miscellaneous flags.
                0000  ;
                0000
      00000000  0000  MF$V_INIT_FLAG  =       0                 ; Initialization in progress.
      00000001  0000  MF$V_EXIT_FLAG  =       1                 ; User requested exit.
      00000002  0000  MF$V_INVERSE    =       2                 ; Field is to be displayed in inverse
                0000                                            ;  video.
      00000003  0000  MF$V_RIP        =       3                 ; Indicates that a refresh is in
                0000                                            ;  progress.
      00000004  0000  MF$V_CONT       =       4                 ; Indicates that the display is to be
                0000                                            ;  updated continuously.
      00000005  0000  MF$V_CYCLE      =       5                 ; Indicates that the display is to be
                0000                                            ;  cycled every other update.
      00000006  0000  MF$V_IN_OPEN    =       6                 ; Indicates that a command procedure
                0000                                            ;  file is open.
      00000007  0000  MF$V_NEW_CHAR   =       7                 ; Indicates terminal characteristics
                0000                                            ;  have been modified.
      00000008  0000  MF$V_SCROLL     =       8                 ; Indicates that the screen has been
                0000                                            ;  scrolled.
```

```
00000009  0000  MF$V_VALID_MSG     =       9                ; Indicates valid ascii string found in
          0000                                              ;  message buffer.
0000000A  0000  MF$V_CR_FOUND      =       10               ; Indicates that a valid ascii string
          0000                                              ;  followed by a carriage return has
          0000                                              ;  been found in the message buffer.
          0000
00000001  0000  MF$M_INIT_FLAG     =       1@MF$V_INIT_FLAG
00000002  0000  MF$M_EXIT_FLAG     =       1@MF$V_EXIT_FLAG
00000004  0000  MF$M_INVERSE       =       1@MF$V_INVERSE
00000008  0000  MF$M_RIP           =       1@MF$V_RIP
00000010  0000  MF$M_CONT          =       1@MF$V_CONT
00000020  0000  MF$M_CYCLE         =       1@MF$V_CYCLE
00000040  0000  MF$M_IN_OPEN       =       1@MF$V_IN_OPEN
00000080  0000  MF$M_NEW_CHAR      =       1@MF$V_NEW_CHAR
00000100  0000  MF$M_SCROLL        =       1@MF$V_SCROLL
00000200  0000  MF$M_VALID_MSG     =       1@MF$V_VALID_MSG
00000400  0000  MF$M_CR_FOUND      =       1@MF$V_CR_FOUND
          0000
          0000  ;
          0000  ; Scroll flags.
          0000  ;
          0000
00000000  0000  SCR$V_MORE_RT      =       0                ; Data has truncated off right of
          0000                                              ;  screen.
00000001  0000  SCR$V_MORE_LFT     =       1                ; Display has been scrolled left.
00000002  0000  SCR$V_MORE_UP      =       2                ; Display has been scrolled up.
00000003  0000  SCR$V_MORE_DWN     =       3                ; Data has been truncated off the
          0000                                              ;  bottom of the screen.
00000004  0000  SCR$V_MORE         =       4                ; Any "more" flag has been set.
          0000
00000001  0000  SCR$M_MORE_RT      =       1@SCR$V_MORE_RT
00000002  0000  SCR$M_MORE_LFT     =       1@SCR$V_MORE_LFT
00000004  0000  SCR$M_MORE_UP      =       1@SCR$V_MORE_UP
00000008  0000  SCR$M_MORE_DWN     =       1@SCR$V_MORE_DWN
00000010  0000  SCR$M_MORE         =       1@SCR$V_MORE
          0000
          0000
00000010  0000  MAX_ID             =       16               ; Maximum number of system that can be
          0000                                              ;  removed from the display using
          0000                                              ;  REMOVE SYSTEM/ID=system_id.
00000002  0000  MAXPDT             =       2                ; Maximum number of port descriptor
          0000                                              ; tables per system.
00000003  0000  QUE_LIMIT          =       3                ; Maximum number of entries to check
          0000                                              ;  in an interlocked queue.
00000012  0000  DEF_CONNCNT        =       18               ; Default value for the SYSGEN
          0000                                              ;  parameter SCSCONNCNT.
          0000
          0000                  .SBTTL  Ascii strings.
```

```
                                  00000000                .PSECT  RODATA,RD,NOWRT,NOEXE
                                      0000
                                      0000    TIME_SOURCE:
                                00    0000            .BYTE   0
                                      0001
                                      0001    ;
                                      0001    ; Filespecs and logical names for initialization.
                                      0001    ;
                                      0001
                                      0001    DEF_EXT:                                        ; Default file extention for
                       49 4E 49 2E    0001            .ASCII  /.INI/                          ;  initialization if SHOW_CLUSTER$INIT
                                      0005                                                    ;  is defined.
                          00000004    0005    DEF_EXT_LEN      =.-DEF_EXT                     ; Length of string.
                                      0005
                                      0005    DEF_NAME2:                                      ; Default command procedures filename
52 45 54 53 55 4C 43 5F 57 4F 48 53   0005            .ASCII  /SHOW_CLUSTER$INIT:.COM/;  if SHOW_CLUSTER$INIT is defined.
   4D 4F 43 2E 3A 54 49 4E 49 24      0011
                          00000016    001B    DEF_NAME2_LEN    =.-DEF_NAME2                   ; Length of string.
                                      001B
                                      001B    ;
                                      001B    ; Names of qualifiers to the DCL level SHOW command.
                                      001B    ;
                                      001B
                                      001B    REPORT_QUAL:                                    ; Qualifier for specifying report name.
54 52 4F 50 45 52 00000023'010E0000'  001B            .ASCID  /REPORT/
                                      0029
                                      0029    CONTINUOUS_QUAL:                                ; Qualifier for specifying continuous
4E 49 54 4E 4F 43 00000031'010E0000'  0029            .ASCID  /CONTINUOUS/                    ;  update display.
                       53 55 4F 55    0037
                                      003B
                                      003B    INTERVAL_QUAL:                                  ; Qualifier for specifying interval
56 52 45 54 4E 49 00000043'010E0000'  003B            .ASCID  /INTERVAL/                      ;  between display updates.
                             4C 41    0049
                                      004B
                                      004B    OUTPUT_QUAL:                                    ; Qualifier for specifying output file.
54 55 50 54 55 4F 00000053'010E0000'  004B            .ASCID  /OUTPUT/                        ;
                                      0059
                                      0059    ENDING_QUAL:                                    ; Qualifier for specifying ending time.
47 4E 49 44 4E 45 00000061'010E0000'  0059            .ASCID  /ENDING/                        ;
                                      0067
                                      0067    BEGIN_QUAL:                                     ; Qualifier for specifying starting
4E 4E 49 47 45 42 0000006F'010E0000'  0067            .ASCID  /BEGINNING/                     ;  time.
                          47 4E 49    0075
                                      0078
                                      0078    ;
                                      0078    ; Ascii strings used to make up the scrolling signals.
                                      0078    ;
                                      0078
                                      0078    MORE_DSCR:
20 20 20 09 09 09 00000080'010E0000'  0078            .ASCID  /                       More    /
   20 20 20 20 65 72 6F 4D 20 20      0086
                                      0090    MORE_TABS:                                      ; Extra tabs for 132 column mode.
09 09 09 09 09 09 00000098'010E0000'  0090            .ASCID  /                                       More    /
      20 20 20 20 65 72 6F 4D 09      009E
          20 20 5E 000000AF'010E0000' 00A7    MORE_UP:.ASCID  /^  /                           ; Scroll down allowed.
                                      00B2    MORE_DWN:
          20 20 76 000000BA'010E0000' 00B2            .ASCID  /v  /                           ; Scroll up allowed.
```

```
                                    00BD  717 MORE_LFT:
        2D 2D 3C 000000C5'010E0000' 00BD  718         .ASCID  /<--/                   ; Scroll left allowed.
        3E 2D 2D 000000D0'010E0000' 00C8  719 MORE_RT:.ASCID  /-->/                   ; Scroll right allowed.
        20 20 20 000000DB'010E0000' 00D3  720 NO_MORE:.ASCID  /   /
                                    00DE  721
                                    00DE  722
                                    00DE  723 ;
                                    00DE  724 ; Logical names.
                                    00DE  725 ;
                                    00DE  726
                                    00DE  727 IN_DEV_NAME:                          ; Input device name.
4E 49 24 53 59 53 000000E6'010E0000' 00DE  728         .ASCID  /SYS$INPUT/             ;
                         54 55 50   00EC
                                    00EF  729
                                    00EF  730 OUT_DEV_NAME:                         ; Output device name.
55 4F 24 53 59 53 000000F7'010E0000' 00EF  731         .ASCID  /SYS$OUTPUT/            ;
                      54 55 50 54   00FD
                                    0101  732
                                    0101  733 NULL_DEV_NAME:                        ; Null device name.
        3A 4C 4E 00000109'010E0000' 0101  734         .ASCID  /NL:/                   ;
                                    010C  735
                                    010C  736 LIBRARY_NAME:                         ; Name of help library for interactive
45 48 24 53 59 53 00000114'010E0000' 010C  737         .ASCID  /SYS$HELP:SHWCLHELP.HLB/;  parser.
50 4C 45 48 4C 43 57 48 53 3A 50 4C 011A
                      42 4C 48 2E   0126
                                    012A  738 INIT_NAME:                            ; Startup initialization file name.
43 5F 57 4F 48 53 00000132'010E0000' 012A  739         .ASCID  /SHOW_CLUSTER$INIT/     ;
   54 49 4E 49 24 52 45 54 53 55 4C 0138
                                    0143  740
                                    0143  741 ;
                                    0143  742 ; Terminal control strings.
                                    0143  743 ;
                                    0143  744
                                    0143  745 DCRT_NOATR_DSCR:                      ; DECcrt control string to turn
        6D 5B 1B 0000014B'010E0000' 0143  746         .ASCID  <ESC>/[m/               ;  off character attributes.
                                    014E  747
                                    014E  748 DCRT_NARROW_DSCR:                     ; Escape sequence used to place DECcrt
  6C 33 3F 5B 1B 00000156'010E0000' 014E  749         .ASCID  /[?3l/                  ;  in 80 column mode.
                                    015B  750 DCRT_WIDE_DSCR:                       ; Escape sequence used to place DECcrt
  68 33 3F 5B 1B 00000163'010E0000' 015B  751         .ASCID  /[?3h/                  ;  in 132 column mode.
                                    0168  752
                                    0168  753 GRAPHICS_DSGNTR:                      ; Escape sequence designate special
        30 29 1B 00000170'010E0000' 0168  754         .ASCID  /)0/                    ;  graphics and line drawing character
                                    0173  755                                       ;  set into G1 area for DECcrt.
                                    0173  756
                                    0173  757 DELM_DSCR:                            ; Control string to output a
        0F 78 0E 0000017B'010E0000' 0173  758         .ASCID  /x/                     ; vertical bar on a DECcrt.
                           00000003 017E  759 DELM_DSCR_LEN   =       3
                                    017E  760
                                    017E  761 INVRS_DSCR:                           ; Escape sequence used to turn on
     6D 37 5B 1B 00000186'010E0000' 017E  762         .ASCID  /[7m/                   ;  inverse video attribute on a DECcrt.
                                    018A  763
                                    018A  764
                                    018A  765 ;
                                    018A  766 ; Misc messages and ascii strings.
                                    018A  767 ;
                                    018A  768
```

```
                                           018A   769 CLUSTER_MSG:
6F 20 77 65 69 56 00000192'010E0000'       018A   770         .ASCID  /View of Cluster from/
72 66 20 72 65 74 73 75 6C 43 20 66        0198
                                6D 6F      01A4
                                           01A6   771 LOCAL_MSG:
20 6C 61 63 6F 4C 000001AE'010E0000'       01A6   772         .ASCID  /Local ports on/
      6E 6F 20 73 74 72 6F 70              01B4
                                           01BC   773 MSG1_DSCR:
65 74 73 79 73 20 000001C4'010E0000'       01BC   774         .ASCID  / system ID /
               20 44 49 20 6D              01CA
                                           01CF   775 MSG2_DSCR:
65 64 6F 6E 20 20 000001D7'010E0000'       01CF   776         .ASCID  /  node: /
                             20 3A         01DD
                                           01DF   777 BUFF_MSG:
75 62 20 49 43 20 000001E7'010E0000'       01DF   778         .ASCID  / CI buffer descriptors in use./
70 69 72 63 73 65 64 20 72 65 66 66        01ED
2E 65 73 75 20 6E 69 20 73 72 6F 74        01F9
                                           0205   779 FREE_BUFF_MSG:
75 62 20 49 43 20 0000020D'010E0000'       0205   780         .ASCID  / CI buffer descriptors free./
70 69 72 63 73 65 64 20 72 65 66 66        0213
      2E 65 65 72 66 20 73 72 6F 74        021F
                                           0229   781
6E 61 6D 6D 6F 43 00000231'010E0000'       0229   782 PROMPT: .ASCID  /Command > /            ; Command input prompt string.
                       20 3E 20 64         0237
                          0000000A         023B   783 PROMPT_LEN = .-PROMPT-8
                                           023B   784
                                           023B   785
                                           023B   786 ;
                                           023B   787 ; Ascii strings for class headers.
                                           023B   788 ;
                                           023B   789
                                           023B   790 CLASS_NAMES:
         52 45 54 53 55 4C 43 00'          023B   791         .ASCIC  /CLUSTER/               ; Cluster (CLUB) class.
                              07           023B
         53 4D 45 54 53 59 53 00'          0243   792         .ASCIC  /SYSTEMS/               ; Systems class.
                              07           0243
         53 52 45 42 4D 45 4D 00'          024B   793         .ASCIC  /MEMBERS/               ; Members class.
                              07           024B
      53 54 49 55 43 52 49 43 00'          0253   794         .ASCIC  /CIRCUITS/              ; Circuits class.
                              08           0253
53 4E 4F 49 54 43 45 4E 4E 4F 43 00'       025C   795         .ASCIC  /CONNECTIONS/           ; Connections class.
                              0B           025C
      53 52 45 54 4E 55 4F 43 00'          0268   796         .ASCIC  /COUNTERS/              ; Counters class.
                              08           0268
         53 54 49 44 45 52 43 00'          0271   797         .ASCIC  /CREDITS/               ; Credits class.
                              07           0271
53 54 52 4F 50 5F 4C 41 43 4F 4C 00'       0279   798         .ASCIC  /LOCAL_PORTS/           ; Local-ports class.
                              0B           0279
            53 52 4F 52 52 45 00'          0285   799         .ASCIC  /ERRORS/                ; Errors class.
                              06           0285
                                           028C   800
                                           028C   801 ;
                                           028C   802 ; ASCIC strings used by display routines to convert an output value to an ascii
                                           028C   803 ; string.
                                           028C   804 ;
                                           028C   805
                                           028C   806 ;
```

```
                        028C    ; Strings used by routine DISPLAY_CABLE_STATUS to display status of the paths
                        028C    ; and cable (PB$B_CBL_STS, PB$B_PO_STS, and PB$B_P1_STS).
                        028C    ;
                        028C
                        028C    CB_STAT_STRG:
      20 20 2D 20 20 00' 028C           .ASCIC  /  -  /                     ; Paths A and B are bad.
                     05  028C
      20 20 2D 20 41 00' 0292           .ASCIC  /A -  /                     ; Path A is good.
                     05  0292
      42 20 2D 20 20 00' 0298           .ASCIC  /  - B/                     ; Path B is good.
                     05  0298
      42 20 2D 20 41 00' 029E           .ASCIC  /A - B/                     ; Paths A and B are good.
                     05  029E
44 45 53 53 4F 52 43 00' 02A4           .ASCIC  /CROSSED/                   ; Cables are crossed.
                     07  02A4
                         02AC
               00000007  02AC   CB_STAT_STRNG_LEN = 7                       ; Maximum possible string length.
                         02AC
                         02AC   ;
                         02AC   ; Strings used by DISPLAY_LOOPBACK_STATUS to display path loopback status
                         02AC   ; (PDT$B_PO_LBSTS and PDT$B_P1_LBSTS).
                         02AC   ;
                         02AC
                         02AC   LB_STAT_STRG:
                         02AC
      42 20 2D 20 41 00' 02AC           .ASCIC  /A - B/                     ; Loopback tests passed on paths A and
                     05  02AC
                         02B2                                               ;  B.
      20 20 2D 20 41 00' 02B2           .ASCIC  /A -  /                     ; Loopback tests passed on path A.
                     05  02B2
      42 20 2D 20 20 00' 02B8           .ASCIC  /  - B/                     ; Loopback tests passed on path B.
                     05  02B8
      20 20 2D 20 20 00' 02BE           .ASCIC  /  -  /                     ; Loopback tests failed on paths A and
                     05  02BE
                         02C4                                               ;  B.
      20 41 2F 4E 20 00' 02C4           .ASCIC  @ N/A @                     ; Loopback testing is not being done.
                     05  02C4
                         02CA
               00000005  02CA   LB_STAT_STRG_LEN = 5                        ; Maximum possible string length.
                         02CA
                         02CA   ;
                         02CA   ; Strings used by DISPLAY_RPORT_TYP to display the remote port type
                         02CA   ; (PB$L_RPORT_TYP).
                         02CA   ;
                         02CA
                         02CA   RPORT_TYP_STRNG:
      30 38 37 49 43 00' 02CA           .ASCIC  /CI780/                     ; Remote port is a CI780...
                     05  02CA
      30 35 37 49 43 00' 02D0           .ASCIC  /CI750/                     ; ... CI750.
                     05  02D0
      30 35 43 53 48 00' 02D6           .ASCIC  /HSC50/                     ; ... HSC50.
                     05  02D6
                     00' 02DC           .ASCIC  //                          ; ... Reserved.
                     00  02DC
      41 50 49 4C 4B 00' 02DD           .ASCIC  /KLIPA/                     ; ... KL10.
                     05  02DD
         54 4E 49 43 00' 02E3           .ASCIC  /CINT/                      ; ... CI node tester.
```

```
                        04  02E3
                 49 4E 00'  02E8                  .ASCIC  /NI/                    ; ... NI.
                        02  02E8
        48 54 53 53 41 50 00'  02EB               .ASCIC  /PASSTH/                ; ... Pass through mode.
                        06  02EB
                            02F2
                  00000006  02F2     RPORT_TYP_STRNG_LEN = 6                      ; Maximum possible string length.
                            02F2
                            02F2     ;
                            02F2     ; Strings used by DISPLAY_PORT_TYPE to display the port type (UCB$B_DEVTYPE).
                            02F2     ;
                            02F2
                            02F2     DEV_TYP_STRNG:
           30 38 37 49 43 00'  02F2               .ASCIC  /CI780/                 ; Port is a CI780.
                        05  02F2
           30 35 37 49 43 00'  02F8               .ASCIC  /CI750/                 ; Port is a CI750.
                        05  02F8
                            02FE
                  00000005  02FE     DEV_TYP_STRNG_LEN = 5                        ; Maximum possible string length.
                            02FE
                            02FE     ;
                            02FE     ; Strings used by DISPLAY_PORT_STATUS to display the port status (UCB$W_STS).
                            02FE     ;
                            02FE
                            02FE     PORT_STATUS_STRNG:
     45 4E 49 4C 46 46 4F 00'  02FE               .ASCIC  /OFFLINE/               ; Port is offline.
                        07  02FE
        45 4E 49 4C 4E 4F 00'  0306               .ASCIC  /ONLINE/                ; Port is online.
                        06  0306
                            030D
                  00000007  030D     PORT_STATUS_STRNG_LEN = 7                    ; Maximum possible string length.
                            030D
                            030D     ;
                            030D     ; Strings used by DISPLAY_CIR_STATE to display the state of a circuit
                            030D     ; (PB$W_STATE).
                            030D     ;
                            030D
                            030D     PB_STATE_STRNG:
        44 45 53 4F 4C 43 00'  030D               .ASCIC  /CLOSED/                ; Circuit is in the closed state.
                        06  030D
     54 4E 45 53 5F 54 53 00'  0314               .ASCIC  /ST_SENT/               ; Start sent.
                        07  0314
        43 45 52 5F 54 53 00'  031C               .ASCIC  /ST_REC/                ; Start received.
                        06  031C
              4E 45 50 4F 00'  0323               .ASCIC  /OPEN/                  ; Circuit is in the open state.
                        04  0323
     4C 49 41 46 5F 43 56 00'  0328               .ASCIC  /VC_FAIL/               ; Virtual circuit failure in progress.
                        07  0328
                            0330
                  00000007  0330     PB_STATE_STRNG_LEN = 7                       ; Maximum possible string length.
                            0330
                            0330     ;
                            0330     ; Strings used by DISPLAY_RSTATE to display the state of the remote port
                            0330     ; (PB$B_RSTATE).
                            0330     ;
                            0330
```

```
                              0330   894 PB_RSTATE_STRNG:
         54 49 4E 49 4E 55 00' 0330   895         .ASCIC  /UNINIT/                ; Remote port is uninitialized.
                           06  0330
   54 49 4E 49 4E 55 5F 4D 00' 0337   896         .ASCIC  /M_UNINIT/              ; Remote port is in maintenance mode
                           08  0337
                              0340   897                                         ;  and is uninitialized.
            42 41 53 49 44 00' 0340   898         .ASCIC  /DISAB/                 ; Remote port is disabled.
                           05  0340
      42 41 53 49 44 5F 4D 00' 0346   899         .ASCIC  /M_DISAB/               ; Remote port is in maintenance mode
                           07  0346
                              034E   900                                         ;  and is disabled.
               42 41 4E 45 00' 034E   901         .ASCIC  /ENAB/                  ; Remote port is enabled.
                           04  034E
         42 41 4E 45 5F 4D 00' 0353   902         .ASCIC  /M_ENAB/                ; Remote port is in maintenance mode
                           06  0353
                              035A   903                                         ;  and is enabled.
                              035A   904
                     00000008 035A   905 PB_RSTATE_STRNG_LEN = 8                 ; Maximum possible string length.
                              035A   906
                              035A   907 ;
                              035A   908 ; Strings used by DISPLAY_CON_STATE to display the connection state
                              035A   909 ; (CDT$W_STATE).
                              035A   910 ;
                              035A   911
                              035A   912 CDT_STATE_STRNG:
         44 45 53 4F 4C 43 00' 035A   913         .ASCIC  /CLOSED/                ; Connection is in the closed state.
                           06  035A
         4E 45 54 53 49 4C 00' 0361   914         .ASCIC  /LISTEN/                ; Connection is in the listen state.
                           06  0361
               4E 45 50 4F 00' 0368   915         .ASCIC  /OPEN/                  ; Connection is in the open state.
                           04  0368
   4B 43 41 5F 43 53 49 44 00' 036D   916         .ASCIC  /DISC_ACK/              ; Disconnect acknowledged.
                           08  036D
   43 45 52 5F 43 53 49 44 00' 0376   917         .ASCIC  /DISC_REC/              ; Disconnect request received.
                           08  0376
54 4E 45 53 5F 43 53 49 44 00' 037F   918         .ASCIC  /DISC_SENT/             ; Disconnect sent.
                           09  037F
48 43 54 4D 5F 43 53 49 44 00' 0389   919         .ASCIC  /DISC_MTCH/             ; Disconnect match.
                           09  0389
   54 4E 45 53 5F 4E 4F 43 00' 0393   920         .ASCIC  /CON_SENT/              ; Connect request sent.
                           08  0393
      4B 43 41 5F 4E 4F 43 00' 039C   921         .ASCIC  /CON_ACK/               ; Connect request sent & acknowledged.
                           07  039C
      43 45 52 5F 4E 4F 43 00' 03A4   922         .ASCIC  /CON_REC/               ; Connect request received.
                           07  03A4
54 4E 45 53 5F 50 43 43 41 00' 03AC   923         .ASCIC  /ACCP_SENT/             ; Accept request sent.
                           09  03AC
   54 4E 45 53 5F 4A 45 52 00' 03B6   924         .ASCIC  /REJ_SENT/              ; Reject sent.
                           08  03B6
      4C 49 41 46 5F 43 56 00' 03BF   925         .ASCIC  /VC_FAIL/               ; Virtual circuit failed.
                           07  03BF
                              03C7   926
                     00000009 03C7   927 DISPLAY_CON_STATE_LEN = 9               ; Maximum possible string length.
                              03C7   928
                              03C7   929 ;
                              03C7   930 ; Strings used by DISPLAY_SCS_STATE to display the SCS send blocked state
                              03C7   931 ; (CDT$W_BLKSTATE).
```

```
                              03C7   932 ;
                              03C7   933
                              03C7   934 CDT_BLKSTATE_STRNG:
          52 41 45 4C 43 00'  03C7   935         .ASCIC  /CLEAR/         ; Not blocked.
                          05  03C7
    44 4E 45 50 5F 4E 4F 43 00'  03CD   936         .ASCIC  /CON_PEND/      ; Waiting to send connection request.
                          08  03CD
 44 4E 45 50 5F 50 43 43 41 00'  03D6   937         .ASCIC  /ACCP_PEND/     ; Waiting to send accept request.
                          09  03D6
    44 4E 45 50 5F 4A 45 52 00'  03E0   938         .ASCIC  /REJ_PEND/      ; Waiting to send reject request.
                          08  03E0
 44 4E 45 50 5F 43 53 49 44 00'  03E9   939         .ASCIC  /DISC_PEND/     ; Waiting to send disconnect request.
                          09  03E9
       44 4E 45 50 5F 52 43 00'  03F3   940         .ASCIC  /CR_PEND/       ; Waiting to send credit.
                          07  03F3
    44 4E 45 50 5F 52 43 44 00'  03FB   941         .ASCIC  /DCR_PEND/      ; Waiting to send credit in
                          08  03FB
                              0404   942                                 ;  preparation for disconnect.
                              0404   943
                    00000009  0404   944 DISPLAY_SCS_STATE_LEN = 9       ; Maximum possible string length.
                              0404   945
                              0404   946 ;
                              0404   947 ; Strings used to display quorum file states.
                              0404   948 ;
                              0404   949
                              0404   950 YES_NO_STRING:
                              0404   951
                 53 45 59 00'  0404   952         .ASCIC  /YES/
                          03  0404
                    4F 4E 00'  0408   953         .ASCIC  /NO/
                          02  0408
                              040B   954
                    00000003  040B   955 YES_NO_STRNG_LEN = 3            ; Maximum length of string.
                              040B   956
                              040B   957 ;
                              040B   958 ; Strings used by DISPLAY_CSB_STATE to display the cluster system block state
                              040B   959 ; (CSB$B_STATE).
                              040B   960 ;
                              040B   961
                              040B   962 CSB_STATE_STRNG:
              4E 45 50 4F 00'  040B   963         .ASCIC  /OPEN/          ; Open.
                          04  040B
        44 45 53 4F 4C 43 00'  0410   964         .ASCIC  /CLOSED/        ; Closed.
                          06  0410
 54 43 45 4E 4E 4F 43 45 52 00'  0417   965         .ASCIC  /RECONNECT/     ; Attempting to reconnect.
                          09  0417
                 57 45 4E 00'  0421   966         .ASCIC  /NEW/           ; New block.
                          03  0421
       54 43 45 4E 4E 4F 43 00'  0425   967         .ASCIC  /CONNECT/       ; Attempting initial connection.
                          07  0425
          54 50 45 43 43 41 00'  042D   968         .ASCIC  /ACCEPT/        ; Accepting initial connection.
                          06  042D
54 43 45 4E 4E 4F 43 53 49 44 00'  0434   969         .ASCIC  /DISCONNECT/    ; Disconnect in progress.
                          0A  0434
    54 50 45 43 43 41 45 52 00'  043F   970         .ASCIC  /REACCEPT/      ; Accepting reconnect request.
                          08  043F
              54 49 41 57 00'  0448   971         .ASCIC  /WAIT/          ; Time-out in progress.
```

```
                              04  0448
             44 41 45 44 00'  044D              .ASCIC  /DEAD/                  ; No connection possible.
                              04  044D
                                  0452
                    0000000A      0452  CSB_STATE_STRNG_LEN = 10                ; Maximum possible length of string.
                                  0452
                                  0452  ;
                                  0452  ; Strings used by DISPLAY_CSB_STATUS to display the status of the node in the
                                  0452  ; cluster (CSB$L_STATUS).
                                  0452  ;
                                  0452
                                  0452  CSB_STATUS_STRNG:
                57 45 4E 00'      0452          .ASCIC  /NEW/                   ; New system in cluster.
                              03  0452
    57 45 4E 5F 4B 52 42 00'      0456          .ASCIC  /BRK_NEW/               ; New system - break in connection.
                              07  0456
       52 45 42 4D 45 4D 00'      045E          .ASCIC  /MEMBER/                ; System is a member of the cluster.
                              06  045E
    4D 45 4D 5F 4B 52 42 00'      0465          .ASCIC  /BRK_MEM/               ; Member - break in connection.
                              07  0465
                4E 4F 4E 00'      046D          .ASCIC  /NON/                   ; System is not a member of the cluster.
                              03  046D
    4E 4F 4E 5F 4B 52 42 00'      0471          .ASCIC  /BRK_NON/               ; Non-member - break in connection.
                              07  0471
    44 45 56 4F 4D 45 52 00'      0479          .ASCIC  /REMOVED/               ; System is removed from the cluster.
                              07  0479
    4D 45 52 5F 4B 52 42 00'      0481          .ASCIC  /BRK_REM/               ; Removed system - break in connection.
                              07  0481
                                  0489
                    00000007      0489  CSB_STATUS_STRNG_LEN = 7                ; Maximum possible string length.
                                  0489
                                  0489  ;
                                  0489  ; Symbol prefix strings used by DISPLAY_FIELD to determine which SCS data
                                  0489  ; structure contains the data.  Prefixes must be less than 4 characters.
                                  0489  ;
                                  0489
                                  0489  SYM_PRFX:
             42 55 4C 43 00'      0489          .ASCIC  /CLUB/                  ; Prefix for club offsets.
                              04  0489
                24 42 53 00'      048E          .ASCIC  /SB$/                   ; Prefix for system block offsets.
                              03  048E
             24 42 53 43 00'      0492          .ASCIC  /CSB$/                  ; Prefix for cluster system block
                              04  0492
                                  0497                                          ;  offsets.
                24 42 50 00'      0497          .ASCIC  /PB$/                   ; Prefix for path block offsets.
                              03  0497
             24 54 44 43 00'      049B  CDT_PFX:.ASCIC  /CDT$/                  ; Prefix for connection descriptor
                              04  049B
                                  04A0                                          ;  table offsets.
             24 54 44 50 00'      04A0  PDT_PFX:.ASCIC  /PDT$/                  ; Prefix for port descriptor table
                              04  04A0
                                  04A5                                          ;  offsets.
                24 42 4D 00'      04A5          .ASCIC  /MB$/                   ; Prefix for miscellaneous buffer
                              03  04A5
                                  04A9                                          ;  offsets.
                                  04A9
                    00000007      04A9  PFX_LIM = 7                             ; Number of valid prefixes.
```

```
                               04A9
                               04A9
                               04A9  ;
                               04A9  ; Table used to convert SCS data size in bytes (FD$B_LENGTH) to number of
                               04A9  ; columns needed to display the data (FD$B_MAX_DATA_SIZE).
                               04A9  ;
                               04A9
                               04A9  CVT_TABLE:
     14 11 0F 0D 0A 08 05 03 00 04A9          .BYTE   0,3,5,8,10,13,15,17,20
                               04B2
                               04B2  ;
                               04B2  ; Table used to convert SB$L_HW_TYPE into a bit value to determine if
                               04B2  ; the type of system should be displayed.
                               04B2  ;
                               04B2
                               04B2  ASSUME  CLS$V_750_SYS    EQ      16
                               04B2  ASSUME  CLS$V_780_SYS    EQ      CLS$V_750_SYS+1
                               04B2  ASSUME  CLS$V_785_SYS    EQ      CLS$V_780_SYS+1
                               04B2  ASSUME  CLS$V_790_SYS    EQ      CLS$V_785_SYS+1
                               04B2  ASSUME  CLS$V_HSC50_SYS  EQ      CLS$V_790_SYS+1
                               04B2  ASSUME  CLS$V_HSC60_SYS  EQ      CLS$V_HSC50_SYS+1
                               04B2
                               04B2  HW_TYPE_TABLE:
                      30353756 04B2          .LONG   ^A/V750/                ; 750 systems.
                      30383756 04B6          .LONG   ^A/V780/                ; 780 systems.
                      35383756 04BA          .LONG   ^A/V785/                ; 785 systems.
                      30303838 04BE          .LONG   ^A/8800/                ; 790 systems.
                      30355348 04C2          .LONG   ^A/HS50/                ; HSC50 systems.
                      30365348 04C6          .LONG   ^A/HS60/                ; HSC60 systems.
                      00000006 04CA  NUM_HW_TYPES =  <.-HW_TYPE_TABLE>/4
                               04CA
                               04CA  ;
                               04CA  ; Table used for Hex to decimal conversion.
                               04CA  ;
                               04CA
             8AC72304 89E80000 04CA  DEC_TAB:.QUAD   10000000000000000000
             0DE0B6B3 A7640000 04D2          .QUAD   1000000000000000000
             01634578 5D8A0000 04DA          .QUAD   100000000000000000
             002386F2 6FC10000 04E2          .QUAD   10000000000000000
             00038D7E A4C68000 04EA          .QUAD   1000000000000000
             00005AF3 107A4000 04F2          .QUAD   100000000000000
             00000918 4E72A000 04FA          .QUAD   10000000000000
             000000E8 D4A51000 0502          .QUAD   1000000000000
             00000017 4876E800 050A          .QUAD   100000000000
             00000002 540BE400 0512          .QUAD   10000000000
             00000000 3B9ACA00 051A          .QUAD   1000000000
             00000000 05F5E100 0522          .QUAD   100000000
             00000000 00989680 052A          .QUAD   10000000
             00000000 000F4240 0532          .QUAD   1000000
             00000000 000186A0 053A          .QUAD   100000
             00000000 00002710 0542          .QUAD   10000
             00000000 000003E8 054A          .QUAD   1000
             00000000 00000064 0552          .QUAD   100
             00000000 0000000A 055A          .QUAD   10
             00000000 00000001 0562          .QUAD   1
                               056A
                      00000014 056A  DEC_TAB_LEN = .-DEC_TAB/8
```

```
                           056A  DEC_TAB_END:
                           056A
                           056A  ;
                           056A  ; Starting and ending addresses of pages to be locked in working set.
                           056A  ; This includes the fixed length snapshot buffers and the snapshot routine
                           056A  ; itself.  It does not include the variable length buffers which must be
                           056A  ; locked separately.
                           056A  ;
                           056A
                           056A  SNAP_ADRS:
                           056A          .ADDRESS -                     ; Starting and ending addresses of
      00003333'00000000'   056A                  SNAP_BUF,SNAP_END      ;  snapshot routine.
                           0572
                           0572  ;
                           0572  ; Item list for terminal QIO function modifiers.
                           0572  ;
                           0572
                           0572  TERM_LIST:
                 0000      0572          .WORD   0
                 0000      0574          .WORD   TRM$_MODIFIERS
             00008000      0576          .LONG   TRM$M_TM_NOEDIT
             00000000      057A          .LONG   0
             0000000C      057E  TERM_LIST_LEN = .-TERM_LIST
                           057E
                           057E  ;
                           057E  ; List of parameters request by $GETDVI.
                           057E  ;
                           057E
                           057E  GET_LIST:
                 0004      057E          .WORD   4                      ; Buffer size.
                 0004      0580          .WORD   DVI$_DEVCLASS          ; Request device class
                           0582          .ADDRESS -                     ; Buffer address.
            00002208'      0582                  DEVICE_CLASS           ;
             00000000      0586          .LONG   0                      ; Length not needed.
                           058A
                 0004      058A          .WORD   4                      ; Buffer size.
                 0006      058C          .WORD   DVI$_DEVTYPE           ; Request device type.
                           058E          .ADDRESS -                     ; Buffer address.
            0000220C'      058E                  DEVICE_TYPE            ;
             00000000      0592          .LONG   0                      ; Length not needed.
                           0596
                 0004      0596          .WORD   4                      ; Buffer size.
                 000A      0598          .WORD   DVI$_DEVDEPEND         ; Request device dependent information.
                           059A          .ADDRESS -                     ; Buffer address.
            00002218'      059A                  DEVDEPEND              ;
             00000000      059E          .LONG   0                      ; Length not needed.
                           05A2
                 0004      05A2          .WORD   4                      ; Buffer size.
                 001C      05A4          .WORD   DVI$_DEVDEPEND2        ; Request additional device dependent
                           05A6                                         ;  information.
                           05A6          .ADDRESS -                     ; Buffer address.
            0000221C'      05A6                  DEVDEPEND2             ;
             00000000      05AA          .LONG   0                      ; Length not needed.
                           05AE
                 0004      05AE          .WORD   4                      ; Buffer size.
                 0008      05B0          .WORD   DVI$_DEVBUFSIZ         ; Request device buffer size.
                           05B2          .ADDRESS -                     ; Buffer address.
```

```
00002210' 05B2  1126                    BUFFER_SIZE
00000000  05B6  1127            .LONG   0                       ; Length not needed.
          05BA  1128
00000000  05BA  1129            .LONG   0                       ; End of list.
          05BE  1130
          05BE  1131
          05BE  1132 ;
          05BE  1133 ; List of parameters request by $GETJPI.
          05BE  1134 ;
          05BE  1135
          05BE  1136 JPI_LIST:
    0004  05BE  1137            .WORD   4                       ; Buffer size.
   0000'  05C0  1138            .WORD   JPI$_CURPRIV            ; Request process's current privileges.
          05C2  1139            .ADDRESS -                      ; Buffer address.
00002230' 05C2  1140                    CUR_PRIV                ;
00000000  05C6  1141            .LONG   0                       ; Length not needed.
          05CA  1142
00000000  05CA  1143            .LONG   0                       ; End of list.
          05CE  1144
          05CE  1145
          05CE  1146            .SBTTL  Local storage.
```

```
          00000000            .PSECT  RWDATA,RD,WRT,NOEXE
                    0000
          00000000  0000  BUFADR: .LONG   0                          ; Temporary storage for queue buffer
                    0004                                             ;  address.
          00000100' 0004  BUFSIZE:.LONG   MBX_BUF_LEN                ; Buffer size.
                    0008
                    0008  MSG_FRE_QUE:                               ; Queue of allocated message buffers.
 00000008'00000008' 0008          .LONG   MSG_FRE_QUE,MSG_FRE_QUE    ;
                    0010  MSG_OUT_QUE:                               ; Queue of broadcast messages to be
 00000010'00000010' 0010          .LONG   MSG_OUT_QUE,MSG_OUT_QUE    ;  output.
                    0018
                    0018  SCRATCH:                                   ; Temporary storage used for displaying
          00000000  0018          .LONG   0                          ;  PROTOCOL field.
                    001C
                    001C  ;
                    001C  ; File access block for input file.  Used when executing a command procedure
                    001C  ; or when executing Show Cluster from a file stream.  The default filename
                    001C  ; extention is .COM.
                    001C  ;
                    001C
                    001C  INFAB:  $FAB    DNM = <.COM>,-
                    001C                  FNA = IN_FILE_SPEC,-
                    001C                  FNS = IN_FILE_SPEC_LEN,-
                    001C                  FOP = NAM,-
                    001C                  NAM = INAM_BLOCK
                    006C
                    006C  INAM_BLOCK:
                    006C          $NAM    ESA = IN_EXP_NAME,-        ; Address to store expanded filespec.
                    006C                  ESS = IN_EXP_NAME_LEN      ; Address to store length of expanded
                    00CC                                             ;  filespec.
                    00CC
                    00CC  IN_FILE_SPEC:                              ; Buffer for filespec for procedure.
          0000011C  00CC          .BLKB   80                         ;
          00000050  011C  IN_FILE_SPEC_LEN = .-IN_FILE_SPEC          ;
                    011C
                    011C  IN_EXP_DSCR:                               ; Descriptor for expanded filespec.
          00000050  011C          .LONG   80                         ;
          00000124' 0120          .LONG   IN_EXP_NAME                ;
                    0124  IN_EXP_NAME:                               ; Buffer for expanded filespec.
          00000174  0124          .BLKB   80                         ;
          00000050  0174  IN_EXP_NAME_LEN = . -IN_EXP_NAME           ;
                    0174
                    0174
                    0174  ;
                    0174  ; Output file access block.  Used for /OUTPUT= or when SYS$OUTPUT has been
                    0174  ; defined to be a file.  The default filename is SHWCLSTR.DAT.
                    0174  ;
                    0174
                    0174  OUTFAB: $FAB    DNM = <SHWCLSTR.DAT>,-
                    0174                  FAC = PUT,-
                    0174                  FNA = OUT_FILE_SPEC,-
                    0174                  FNS = OUT_FILE_SPEC_LEN,-
                    0174                  FOP = NAM,-
                    0174                  NAM = OUTNAM_BLOCK
                    01C4
                    01C4  OUTNAM_BLOCK:
                    01C4          $NAM    ESA = OUT_EXP_NAME,-       ; Address to store expanded filespec.
```

```
         01C4                  ESS = OUT_EXP_NAME_LEN  ; Address to store length of expanded
         0224                                          ;  filespec.
         0224
         0224  OUT_FILE_SPEC:                          ; Buffer for filespec for procedure.
00000274 0224          .BLKB   80                      ;
00000050 0274  OUT_FILE_SPEC_LEN = .-OUT_FILE_SPEC     ;
         0274
         0274  OUT_EXP_DSCR:                           ; Descriptor for expanded filespec.
00000050 0274          .LONG   80                      ;
0000027C' 0278         .LONG   OUT_EXP_NAME            ;
         027C  OUT_EXP_NAME:                           ; Buffer for expanded filespec.
000002CC 027C          .BLKB   80                      ;
00000050 02CC  OUT_EXP_NAME_LEN = . -OUT_EXP_NAME      ;
         02CC
         02CC  ;
         02CC  ; Establish a record access block.  Used when inputing from a procedure or
         02CC  ; when executing from a file stream.
         02CC  ;
         02CC
         02CC  INRAB:  $RAB    FAB = INFAB,-           ; Pointer to FAB.
         02CC                  UBF = REC_BUFF,-        ; Record buffer.
         02CC                  USZ = REC_SIZE          ; Record buffer size.
         0310
         0310  REC_DSCR:                               ; Descriptor for record buffer.
00000000 0310          .LONG   0                       ;
00000318' 0314         .LONG   REC_BUFF                ;
         0318  REC_BUFF:                               ; Record buffer.
0000039C 0318          .BLKB   132                     ;
00000084 039C  REC_SIZE = .-REC_BUFF                   ;
         039C
         039C
         039C  TRAN_DSCR:                              ; Descriptor for SHOW_CLUSTER$INIT
         039C                                          ;  translation.
00000040 039C          .LONG   64                      ; Length.
         03A0          .ADDRESS -                      ; Address of buffer.
000003A4' 03A0                 TRAN_BUF                ;
         03A4  TRAN_BUF:                               ; Buffer.
000003E4 03A4          .BLKB   64                      ;
         03E4
         03E4  BUF_COUNT:                              ; Number of times we've allocated some
      00 03E4          .BYTE   0                       ;  buffer space. (2 max)
         03E5
         03E5  SB_COUNT:                               ; Maximum number of entries to be stored
    0000 03E5          .WORD   0                       ;  in the system block buffer.
         03E7  CSB_COUNT:                              ; Maximum number of entries to be stored
    0000 03E7          .WORD   0                       ;  in the cluster system block buffer.
         03E9  PB_COUNT:                               ; Maximum number of entries to be stored
    0000 03E9          .WORD   0                       ;  in the path block buffer.
         03EB  CDT_COUNT:                              ; Maximum number of entries to be stored
    0000 03EB          .WORD   0                       ;  in the CDT buffer.
         03ED
00000000 03ED  VAR_BUF:.LONG   0                       ; Address of variable length buffer
         03F1                                          ;  returned from GET_VM.
         03F1  VAR_BUF_END:                            ; End of variable buffer space.
00000000 03F1          .LONG   0                       ;
         03F5  VAR_BUF_LEN:                            ; Length of variable buffer.
00000000 03F5          .LONG   0                       ;
```

```
                   03F9
                   03F9  ;
                   03F9  ; Descriptors for variable length buffers from previous snapshot.
                   03F9  ;
                   03F9
                   03F9  OLD_SB_DSCR:                           ; System block buffer.
00000000 00000000  03F9          .QUAD   0                      ;
                   0401  OLD_CSB_DSCR:                          ; Cluster system block buffer.
00000000 00000000  0401          .QUAD   0                      ;
                   0409  OLD_PB_DSCR:                           ; Path block buffer.
00000000 00000000  0409          .QUAD   0                      ;
                   0411  OLD_CDT_DSCR:                          ; CDT buffer.
00000000 00000000  0411          .QUAD   0                      ;
                   0419
                   0419  OLD_VAR_BUF:                           ; Address of previous buffer.
         00000000  0419          .LONG   0                      ;
                   041D
                   041D  OLD_VAR_BUF_END:                       ; End of previous buffer.
         00000000  041D          .LONG   0                      ;
                   0421
                   0421  OLD_VAR_BUF_LEN:                       ; Length of previous buffer.
         00000000  0421          .LONG   0                      ;
                   0425
                   0425  NUM_FIELDS:                            ; Number of fields to be displayed
               00  0425          .BYTE   0                      ;  Used to limit left-scrolling.
                   0426  SCS_NUM_FIELDS:                        ; Number of SCS fields to be displayed.
               00  0426          .BYTE   0                      ;
                   0427  CLUB_NUM_FIELDS:                       ; Number of CLUB fields to be displayed.
               00  0427          .BYTE   0                      ;
                   0428
                   0428  NUM_LINES:                             ; Number of lines of data this pass.
         00000000  0428          .LONG   0                      ;
                   042C  PRIOR_NUM_LINES:                       ; Number of lines of data last pass.
         00000000  042C          .LONG   0                      ;
                   0430  SCROLL_LINES:                          ; Copy of PRIOR_NUM_LINES used to
         00000000  0430          .LONG   0                      ;  limit up scrolling.
                   0434
                   0434  ;
                   0434  ; LINE_DIF is a record of the lines in the display that need to be redrawn
                   0434  ; because data on that line has changed.  Bit 7 corresponds to line #7,
                   0434  ; Bit 8 - line #8, etc.  LINE_DIF is used to save the time of redrawing lines
                   0434  ; in the display that haven't changed since the last pass.
                   0434  ;
                   0434  ; PRIOR_LINE_DIF is a record of the lines that were redrawn in the last pass.
                   0434  ; PRIOR_LINE_DIF is used to turn off fields that have been set to inverse
                   0434  ; video on the previous pass.
                   0434  ;
                   0434  ; Differences in lines are detected by the following routines:
                   0434  ;
                   0434  ;       GET_DATA        (Detects changes in data through setting of inverse
                   0434  ;                       video flag by the CHECK_DIF routine)
                   0434  ;       CHECK_POS       Detects change in area of the screen where a system
                   0434  ;                       is displayed without the data changing.
                   0434  ;       CHECK_NUM_PB    Detects changes in the number of circuits so that
                   0434  ;                       NUM_CIRCUITS is updated.
                   0434  ;       CHECK_NUM_CDT   Detects changes in the number of connections so that
                   0434  ;                       NUM_CONNECTIONS is updated.
                   0434  ;
```

```
          0434  ;
          0434  ; NOTES:
          0434  ;
          0434  ;        LINE_DIF is cleared (actually SCROLL_DIF is copied to LINE_DIF)
          0434  ;        before each display is built after being copied to PRIOR_LINE_DIF.
          0434  ;
          0434  ;        LINE_DIF is used only by GET_DATA to determine if the line has to
          0434  ;        be redrawn because the line has fields that need to be updated.
          0434  ;
          0434  ;        PRIOR_LINE_DIF is used only by GET_DATA to redrawn the line so that
          0434  ;        any fields set to inverse video will be cleared.
          0434  ;
          0434
          0434  LINE_DIF:                               ; Reserve one longword to record the
00000000  0434          .LONG   0                       ; changes in data lines on the current
          0438                                          ;  pass.
          0438  PRIOR_LINE_DIF:                         ; Reserve one longword to save a record
00000000  0438          .LONG   0                       ; of changes in data lines on the
          043C                                          ;  previous pass.
          043C
          043C  ;
          043C  ; BLANKS_LINES is used to record which lines in the display have all the fields
          043C  ; filled in with blanks.  This together with PRIOR_BLANK_LINES is used to
          043C  ; save the time of redrawing a blank line every time the screen is updated.
          043C  ;
          043C  ;NOTES:
          043C  ;
          043C  ;        BLANK_LINES is cleared before each display is build after being
          043C  ;        copied to PRIOR_BLANK_LINES.
          043C  ;
          043C  ;        BLANK_LINES and PRIOR_BLANK_LINES are used only by the GET_BLANK
          043C  ;        routine to determine if the line should be drawn or not.
          043C  ;
          043C
          043C  BLANK_LINES:                            ; Reserve one longword to save a record
00000000  043C          .LONG   0                       ;  of the blank lines that are output.
          0440  PRIOR_BLANK_LINES:                      ; Reserve one longword to save a record
00000000  0440          .LONG   0                       ;  of blank lines from previous pass.
          0444
          0444  ;
          0444  ; SCROLL_DIF is used to record which lines is the display must be redrawn
          0444  ; because the screen was hardware scrolled.  When the screen is hardware
          0444  ; scrolled one or more blank lines are left at the top or bottom of the
          0444  ; scrolling region and need to be filled in.
          0444  ;
          0444  ;NOTES:
          0444  ;
          0444  ;        SCROLL_DIF is cleared after the display has been built and before
          0444  ;        any scroll commands can be issued.
          0444  ;
          0444  ;        SCROLL_DIF is copied to LINE_DIF before the display is built so
          0444  ;        that the line are considered ''changed'' lines.
          0444  ;
          0444
          0444  SCROLL_DIF:                             ; Indicates which line on a DECcrt
00000000  0444          .LONG   0                       ;  must be redrawn because the screen
          0448                                          ;  scrolled.  (Because the screen is
```

```
                 0448  1376                                              ;  hardware scrolled).
                 0448  1377
                 0448  1378 NODE_NAME_DSCR:                              ; Reserve room for storing the node
        00000000  0448  1379          .LONG   0                          ;  name of the local system.
       00000450'  044C  1380          .LONG   NODE_NAME                  ;
                 0450  1381 NODE_NAME:                                   ;
        00000458  0450  1382          .BLKB   8                          ;
                 0458  1383
                 0458  1384 ;
                 0458  1385 ; Scrolling parameters.
                 0458  1386 ;
                 0458  1387
                 0458  1388 SCROLL_DELAY:                                ; Number of refreshes it takes to
        00000000  0458  1389          .LONG   0                          ;  scroll the bottom two lines out of
                 045C  1390                                              ;  the scrolling region.
                 045C  1391
        00000000  045C  1392 V_SCROLL:.LONG  0                           ; Number of data lines to be skipped by
                 0460  1393                                              ;  the GET_DATA routine before a line
                 0460  1394                                              ;  of data is added to the screen
                 0460  1395                                              ;  buffer.
                 0460  1396 V_SCROLL_LINE:                               ; Counter for the number of lines
        00000000  0460  1397          .LONG   0                          ;  skipped by the GET_DATA routine.
                 0464  1398
                 0464  1399 CLSTR_H_SCROLL:                              ; Number of field entries to be skipped
        00000000  0464  1400          .LONG   0                          ;  in the cluster report by the
                 0468  1401                                              ;  FIND_ENTRY routine before success is
                 0468  1402                                              ;   returned.
                 0468  1403
                 0468  1404 LOCAL_H_SCROLL:                              ; Number of field entries to be skipped
        00000000  0468  1405          .LONG   0                          ;  in the local report by the
                 046C  1406                                              ;  FIND_ENTRY routine before success is
                 046C  1407                                              ;  returned.
                 046C  1408
                 046C  1409 H_SCROLL_LINE:                               ; Counter for the number of entries
        00000000  046C  1410          .LONG   0                          ;  skipped by the FIND_ENTRY routine.
                 0470  1411
                 0470  1412 TOP_CMD_LINE:                                ; Indicates type of information last
              00  0470  1413          .BYTE   0                          ;  output on top command line.
                 0471  1414
                 0471  1415 BOT_CMD_LINE:                                ; Indicates type of information last
              00  0471  1416          .BYTE   0                          ;  output on bottom command line.
                 0472  1417
                 0472  1418
                 0472  1419 ;
                 0472  1420 ; Flags.
                 0472  1421 ;
                 0472  1422
                 0472  1423 FLAG_STATE:                                  ; Longword to receive event flag
        00000000  0472  1424          .LONG   0                          ;  status.
                 0476  1425
                 0476  1426 MISC_FLAGS:                                  ; Miscellaneous flags.
        00000000  0476  1427          .LONG   0                          ;
                 047A  1428
                 047A  1429 SCROLL_FLAGS:                                ; Flags to indicate which way the
              00  047A  1430          .BYTE   0                          ;  screen can be scrolled.
                 047B  1431
                 047B  1432 LOCAL_SCROLL_FLAGS:                          ; Used to save a copy of the scroll
```

```
                  00 047B                 .BYTE   0                             ;  flags for the local report.
                     047C
                     047C CLSTR_SCROLL_FLAGS:                                   ; Used to save a copy of the scroll
                  00 047C                 .BYTE   0                             ;  flags for the cluster report.
                     047D
                     047D HELP_FLAGS:                                           ; Flags for LBR$OUTPUT_HELP.
            00000021 047D                 .LONG   HLP$M_PROMPT!HLP$M_HELP       ; Run help in prompting mode.
                     0481                                                       ; With list of topics available.
                     0481
                     0481 ;
                     0481 ; Exit routine.
                     0481 ;
                     0481
                     0481 EXIT_BLK:
            00000000 0481                 .LONG   0                             ; Forward link.
                     0485                 .ADDRESS -                            ; Exit handler address.
           00000F39' 0485                         EXIT_RTN                      ;
            00000001 0489                 .LONG   1                             ; Number of arguments.
                     048D                 .ADDRESS -                            ; Address to receive exit status.
           00000491' 048D                         STATUS                        ;
                     0491
                     0491 STATUS:                                               ; Status code from $EXIT.
            00000000 0491                 .LONG   0                             ;
                     0495
                     0495 ;
                     0495 ; Descriptor for token.  Used when a class name is parsed in case an
                     0495 ; error occurs.
                     0495 ;
                     0495
                     0495 TKN_DSCR:
   00000000 00000000 0495                 .QUAD   0
                     049D
                     049D
                     049D ERR_MSGVEC:
                0001 049D                 .WORD   1                             ; Argument count.
                000F 049F                 .WORD   ^XOF                          ; Message flags.
            00000000 04A1                 .LONG   0                             ; Message identification.
                0001 04A5                 .WORD   1                             ; FAO count.
                0000 04A7                 .WORD   0                             ; New message flags.
                     04A9                 .ADDRESS -                            ; Address of FAO argument.
           000004D7' 04A9                         TPARSE_BLK+TPA$L_TOKENCNT
                     04AD
                     04AD MSG_OUT_ADRS:
            00000000 04AD                 .LONG   0                             ; Address to receive OUTADRS
                     04B1                                                       ;  information from $GETMSG.
                     04B1
                     04B1 ;
                     04B1 ; Broadcast messages.
                     04B1 ;
                     04B1
            00000200 04B1 MAX_MSG:.LONG   512                                   ; Max message length.
                     04B5
            00000200 04B5 BUF_QUO:.LONG   512                                   ; Number of system memory bytes that
                     04B9                                                       ;  can be used to buffer messages.
                     04B9
                     04B9 ;
                     04B9 ; Parsing
```

```
                  04B9 ;
                  04B9
                  04B9 CMD_STATUS:                             ; Status returned from LIB$TPARSE.
         00000000 04B9           .LONG    0                    ;
                  04BD
                  04BD TYPEAHD_BUF:                            ; Buffer to receive type ahead info
00000000 00000000 04BD           .QUAD    0                    ;  from sensemode.
                  04C5
                  04C5 INSTRNG_LEN:                            ; Word to receive length of input string
             0000 04C5           .WORD    0                    ;
                  04C7
                  04C7 TPARSE_BLK:                             ; TPARSE parameter block.
         00000008 04C7           .LONG    TPA$K_COUNT0         ; Longword count.
         00000002 04CB           .LONG    TPA$M_ABBREV         ; Allow abbreviation.
         0000053F 04CF           .BLKL    TPA$K_LENGTH0-8      ; Reserve room for parameters.
                  053F
         00000000 053F KEY_VAL:.LONG      0                    ; Longword to receive keyword value
                  0543                                         ;  from LIB$LOOKUP_KEY.
                  0543
         00000000 0543 KEY_OFF:.LONG      0                    ; Address of longword containing the bit
                  0547                                         ;  which corresponds to the keyword.
                  0547
         00000000 0547 KEY_MSK:.LONG      0                    ; Mask of the bit corresponding to the
                  054B                                         ;  keyword.
                  054B
                  054B KEY_DSCR:                               ; Address of the field descriptor
         00000000 054B           .LONG    0                    ;  corresponding to the keyword.
                  054F
         00000000 054F VALUE:    .LONG    0                    ; Longword to receive value for SET
                  0553                                         ;  FIELD, SET INTERVAL, or SET SCREEN
                  0553                                         ;  command.
                  0553
                  0553 HELP_DSCR:                              ; Descriptor for help keywords.
00000000 00000000 0553           .QUAD    0                    ;
                  055B
                  055B ;
                  055B ; Descriptors for display buffers.
                  055B ;
                  055B
                  055B SCREEN_DSCR:                            ; Descriptor for SCREEN_BUF - the
        00000200' 055B           .LONG    SCREEN_BUF_LEN       ;  buffer for the screen package.
                  055F           .ADDRESS -                    ;
        00000563' 055F                    SCREEN_BUF           ;
                  0563 SCREEN_BUF:                             ;
         00000763 0563           .BLKB    512                  ;
         00000200 0763 SCREEN_BUF_LEN = .-SCREEN_BUF           ;
                  0763
                  0763 CLASS_BOX_DSCR:                         ; Descriptor for CLASS_BOX_BUF - the
        00000200' 0763           .LONG    CLASS_BOX_LEN        ;  buffer for line above the classes in
                  0767           .ADDRESS -                    ;  the header.
        0000076B' 0767                    CLASS_BOX_BUF        ;
                  076B CLASS_BOX_BUF:                          ;
         0000096B 076B           .BLKB    512                  ;
         00000200 096B CLASS_BOX_LEN = .-CLASS_BOX_BUF         ;
                  096B
                  096B FIELD_TOP_BOX_DSCR:                     ; Descriptor for FIELD_TOP_BOX_BUF - the
        00000200' 096B           .LONG    FIELD_TOP_BOX_LEN    ;  buffer for the line above the field
```

```
             096F        .ADDRESS -                          ;  titles in the header.
00000973'    096F                  FIELD_TOP_BOX_BUF         ;
             0973 FIELD_TOP_BOX_BUF:                         ;
00000B73     0973        .BLKB     512                       ;
00000200     0B73 FIELD_TOP_BOX_LEN = .-FIELD_TOP_BOX_BUF    ;
             0B73
             0B73 FIELD_BOT_BOX_DSCR:                        ; Descriptor for FIELD_BOT_BOX_BUF - the
00000200'    0B73        .LONG     FIELD_BOT_BOX_LEN         ;  buffer for the line below the field
             0B77        .ADDRESS -                          ;  title in the header.
00000B7B'    0B77                  FIELD_BOT_BOX_BUF         ;
             0B7B FIELD_BOT_BOX_BUF:                         ;
00000D7B     0B7B        .BLKB     512                       ;
00000200     0D7B FIELD_BOT_BOX_LEN = .-FIELD_BOT_BOX_BUF    ;
             0D7B
             0D7B LAST_LINE_DSCR:                            ; Descriptor for LAST_LINE_BUF - the
00000200'    0D7B        .LONG     LAST_LINE_LEN             ;  buffer for the line following the
             0D7F        .ADDRESS -                          ;  last line of data.
00000D83'    0D7F                  LAST_LINE_BUF             ;
             0D83 LAST_LINE_BUF:                             ;
00000F83     0D83        .BLKB     512                       ;
00000200     0F83 LAST_LINE_LEN = .-LAST_LINE_BUF            ;
             0F83
             0F83 LAST_CLUB_LINE_DSCR:                       ; Descriptor for LAST_CLUB_LINE_BUF -
00000200'    0F83        .LONG     LAST_CLUB_LINE_LEN        ;  the buffer for the line following the
             0F87        .ADDRESS -                          ;  last line of data in the CLUB
00000F8B'    0F87                  LAST_CLUB_LINE_BUF        ;  portion of the CLUSTER report.
             0F8B LAST_CLUB_LINE_BUF:                        ;
0000118B     0F8B        .BLKB     512
00000200     118B LAST_CLUB_LINE_LEN =      .-LAST_CLUB_LINE_BUF
             118B
             118B TITLE_DSCR:                                ; Descriptor for TITLE_BUF - the buffer
00000400'    118B        .LONG     TITLE_BUF_LEN             ;  which receives the titles for
             118F        .ADDRESS -                          ;  display.
00001193'    118F                  TITLE_BUF                 ;
             1193 TITLE_BUF:                                 ;
00001593     1193        .BLKB     1024                      ;
00000400     1593 TITLE_BUF_LEN = .-TITLE_BUF                ;
             1593
             1593 CLASS_DSCR:                                ; Descriptor for CLASS_BUF - the buffer
00000400'    1593        .LONG     CLASS_BUF_LEN             ;  which receives the class names for
             1597        .ADDRESS -                          ;  display.
0000159B'    1597                  CLASS_BUF                 ;
             159B CLASS_BUF:                                 ;
0000199B     159B        .BLKB     1024                      ;
00000400     199B CLASS_BUF_LEN = .-CLASS_BUF                ;
             199B
             199B DATA_DSCR:                                 ; Descriptor for DATA_BUF the buffer
00000400'    199B        .LONG     DATA_BUF_LEN              ;  which receives the data for display.
             199F        .ADDRESS -                          ;
000019A3'    199F                  DATA_BUF                  ;
             19A3 DATA_BUF:                                  ;
00001DA3     19A3        .BLKB     1024                      ;
00000400     1DA3 DATA_BUF_LEN = .-DATA_BUF                  ;
             1DA3
             1DA3
             1DA3 ;
             1DA3 ; Miscellaneous descriptors.
```

```
                1DA3    ;
                1DA3
                1DA3    GET_VAL_DSCR:                                  ; Descriptor for call to CLI$GET_VALUE.
     0000003F'  1DA3            .LONG   RET_BUF_LEN                    ;
                1DA7            .ADDRESS -                             ;
     00001DB3'  1DA7                    RET_BUF                        ;
                1DAB
                1DAB    RET_BUF_DSCR:                                  ; Descriptor for buffer to receive
     0000003F'  1DAB            .LONG   RET_BUF_LEN                    ;  string from CLI$GET_VALUE.
                1DAF            .ADDRESS -                             ;
     00001DB3'  1DAF                    RET_BUF                        ;
     00001DF2   1DB3    RET_BUF:.BLKB   63                             ;
     0000003F   1DF2    RET_BUF_LEN = .-RET_BUF                        ;
                1DF2
                1DF2    LAST_BRDCST_DSCR:                              ; Descriptor for a copy of the last
     00000000   1DF2            .LONG   0                              ;  broadcast message to be used to
                1DF6            .ADDRESS -                             ; scroll the broadcast messages on
     00001DFA'  1DF6                    LAST_BRDCST_BUF                ; a VT52.
                1DFA    LAST_BRDCST_BUF:                               ;
     00001E7E   1DFA            .BLKB   132                            ;
                1E7E
                1E7E
                1E7E    ERR_MSG_DSCR:                                  ; Descriptor used to output an error
     00000000   1E7E            .LONG   0                              ;  message.
                1E82            .ADDRESS -                             ;
     00001E8E'  1E82                    GET_MSG_BUF                    ;
                1E86
                1E86
                1E86    GET_MSG_DSCR:                                  ; Descriptor for buffer for $GETMSG.
     00000100'  1E86            .LONG   GET_MSG_BUF_LEN                ;
                1E8A            .ADDRESS -                             ;
     00001E8E'  1E8A                    GET_MSG_BUF                    ;
                1E8E    GET_MSG_BUF:                                   ;
     00001F8E   1E8E            .BLKB   256                            ;
     00000100   1F8E    GET_MSG_BUF_LEN = .-GET_MSG_BUF                ;
                1F8E
                1F8E
                1F8E    ASC_TIM_DSCR:                                  ; Descriptor used to format INCN_TIME.
     00000017   1F8E            .LONG   23                             ; Enough room for DD-MMM-YYYY HH:MM.
                1F92            .ADDRESS -                             ;
     00001F96'  1F92                    TIMBUF                         ;
     00001FAD   1F96    TIMBUF: .BLKB   23                             ;
                1FAD
                1FAD    MBX_MSG_DSCR:                                  ; Descriptor for mailbox message buffer.
     00000000   1FAD            .LONG   0                              ; Reserved to receive message length.
     00000000   1FB1            .LONG   0                              ; Address of broadcast message (filled
                1FB5                                                   ;  in by OUT_MSG routine).
                1FB5
     000020B5   1FB5    MBX_BUF:.BLKB   256                            ; Buffer for receiving mailbox messages.
     00000100   20B5    MBX_BUF_LEN = .-MBX_BUF                        ;
                20B5
                20B5    TIME_DSCR:                                     ; Descriptor for buffer to receive
     00000014'  20B5            .LONG   TIME_LEN                       ;  current date and time from
                20B9            .ADDRESS -                             ;  LIB$DATE_TIME.
     000020BD'  20B9                            TIME_BUF               ;
     000020D1   20BD    TIME_BUF:.BLKB  20                             ;
     00000014   20D1    TIME_LEN = .-TIME_BUF                          ;
```

```
                     20D1
                     20D1   DDM_SIZE:                               ; Size of the display discription
              0000   20D1           .WORD   0                       ;  message.
                     20D3
                     20D3   CMD_LINE_DSCR:                          ; Descriptor for command input buffer.
         00000078'   20D3           .LONG   CMD_BUF_LEN             ;
                     20D7           .ADDRESS -                      ;
         000020DB'   20D7                   CMD_BUF                 ;
                     20DB   CMD_BUF:                                ;
          00002153   20DB           .BLKB   120                     ;
          00000078   2153   CMD_BUF_LEN = .-CMD_BUF                 ;
                     2153
                     2153   FORMAT_BUF_DSCR:                        ; Descriptor for FORMAT_BUF - used to
                     2153   FORMAT_BUF_LEN:                         ;  store the data after formatting the
          00000000   2153           .LONG   0                       ;  data in ascii.
                     2157           .ADDRESS -                      ;
         0000215B'   2157                   FORMAT_BUF              ;
                     215B   FORMAT_BUF:                             ;
          000021AB   215B           .BLKB   80                      ;
                     21AB
                     21AB
                     21AB   ;
                     21AB   ; Miscellaneous parameters.
                     21AB   ;
                     21AB
                00   21AB   SWITCH: .BYTE   0                       ; Controls switching between reports
                     21AC                                           ;  when /REPORT=CYCLE.  Reports are
                     21AC                                           ;  switched when bit 0 is cleared.
                     21AC
 FFFFFFFF F70F2E80   21AC   DELAY:  .LONG   -10*1000*1000*15,-1     ; Delay time between display updates
                     21B4                                           ;  Defaults to fifteen seconds.
                     21B4
 00000000 00000000   21B4   A_TIME: .QUAD   0                       ; Quadword to receive binary time for
                     21BC                                           ;  /ENDING and /BEGINNING.
                     21BC
                     21BC   TITLE_COUNT:                            ; Used by GET_TITLES to calculate the
          00000000   21BC           .LONG   0                       ;  length of the titles in characters.
                     21C0
                     21C0   TITLE_START:                            ; Used by GET_TITLES to calculate the
          00000000   21C0           .LONG   0                       ;  length of the titles in bytes.
                     21C4
                     21C4   HEADER_SIZE:                            ; Size of the header in characters.
          00000000   21C4           .LONG   0                       ;
                     21C8
                     21C8   CLUB_HEADER_SIZE:                       ; Size of CLUSTER class header.
          00000000   21C8           .LONG   0                       ;
                     21CC
                     21CC   SCS_HEADER_SIZE:                        ; Size of SCS class header.
          00000000   21CC           .LONG   0                       ;
                     21D0   ;
                     21D0   ; Array for storing the length in characters of the titles as calculated by
                     21D0   ; the GET_TITLES and SAVE_TITLE_LEN routines.  This length is used by the
                     21D0   ; GET_CLASSES routines to center the class name.
                     21D0   ;
                     21D0   ;               +--------+--------+--------+--------+
                     21D0   ;               |      Size of CLUSTER titles       |
                     21D0   ;               +--------+--------+--------+--------+
```

```
                  21D0  ;                         |        Size of SYSTEMS titles       |
                  21D0  ;                         +--------+--------+--------+--------+
                  21D0  ;                         |        Size of MEMBERS titles       |
                  21D0  ;                         +--------+--------+--------+--------+
                  21D0  ;                         |        Size of CIRCUITS titles      |
                  21D0  ;                         +--------+--------+--------+--------+
                  21D0  ;                         |      Size of CONNECTIONS titles     |
                  21D0  ;                         +--------+--------+--------+--------+
                  21D0  ;                         |        Size of COUNTERS titles      |
                  21D0  ;                         +--------+--------+--------+--------+
                  21D0  ;                         |        Size of CREDITS titles       |
                  21D0  ;                         +--------+--------+--------+--------+
                  21D0  ;                         |      Size of LOCAL_PORTS titles     |
                  21D0  ;                         +--------+--------+--------+--------+
                  21D0  ;                         |        Size of ERRORS titles        |
                  21D0  ;                         +--------+--------+--------+--------+
                  21D0
                  21D0  TITLE_LEN_ARRAY:
        000021F4  21D0          .BLKL   9
                  21F4
                  21F4  ;
                  21F4  ; Miscellaneous parameters for screen display.
                  21F4  ;
                  21F4
                  21F4  CURRENT_CLASS:                                  ; Indicates which class is currently
        00000000  21F4          .LONG   0                               ;  being processed for display.
                  21F8
                  21F8  DATA_CNTRL:                                     ; Controls which classes will be
        00000000  21F8          .LONG   0                               ;  processed for data display.
                  21FC
                  21FC  SAVED_CLASSES:                                  ; Used to save class selections when
        00000000  21FC          .LONG   0                               ;  generating the display.
                  2200
                  2200  LOCAL_SAVE:                                     ; Used to save local class selections
        00000000  2200          .LONG   0                               ;  when switching to cluster report.
                  2204
                  2204  CLUST_SAVE:                                     ; Used to save cluster report
        00000000  2204          .LONG   0                               ;  selections when switching to the
                  2208                                                  ;  local report.
                  2208
                  2208
                  2208  ;
                  2208  ; Buffer for device information from $GETDVI.
                  2208  ;
                  2208
                  2208  DEVICE_CLASS:                                   ; Device class returned from $GETDVI.
        00000000  2208          .LONG   0                               ;
                  220C  DEVICE_TYPE:                                    ; Device type returned from $GETDVI.
        00000000  220C          .LONG   0                               ;
                  2210  BUFFER_SIZE:                                    ; Width of the terminals screen
        00000000  2210          .LONG   0                               ;  returned from $GETDVI.
                  2214
                  2214  SAVE_CHAR_BUF:
              00  2214          .BYTE   0                               ; Device class.
              00  2215          .BYTE   0                               ; Device type.
            0000  2216          .WORD   0                               ; Screen width.
                  2218  DEVDEPEND:                                      ; Device terminal characteristics
```

```
          00000000  2218          .LONG   0                  ;  returned from $GETDVI.
                    221C DEVDEPEND2:                         ; Extended terminal characteristics
          00000000  221C          .LONG   0                  ;  returned from $GETDVI.
          0000000C  2220 CHAR_BUF_LEN = .-SAVE_CHAR_BUF
                    2220
                    2220
                    2220 NEW_CHAR_BUF:                       ; Reserve space for a copy of
          00002222  2220          .BLKB   2                  ; Device class, type.
                    2222 SCREEN_WIDTH:                       ;
              0000  2222          .WORD   0                  ; Screen width.
          0000222C  2224          .BLKL   2                  ; Characteristics, extended char.
                    222C
                    222C OUTPUT_WIDTH:                       ; Output width for LBR$OUTPUT_HELP.
          00000000  222C          .LONG   0                  ;
                    2230
                    2230
                    2230 ;
                    2230 ; Buffer for process information from $GETJPI.
                    2230 ;
                    2230
                    2230 CUR_PRIV:                           ; Mask of processes current privileges.
          00000000  2230          .LONG   0                  ;
                    2234
                    2234 ;
                    2234 ; Storage area for various terminal parameters.
                    2234 ;
                    2234
              0000  2234 IN_CHAN:.WORD    0                  ; Word to receive channel number for
                    2236                                     ;  terminal input.
                    2236
                    2236 OUT_CHAN:                           ; Word to receive channel number for
              0000  2236          .WORD   0                  ;  terminal output.
                    2238
                    2238 SET_CHAN:                           ; Required by LIB$SET_OUTPUT.
          00000000  2238          .LONG   0                  ;
                    223C
              0000  223C MBXCHAN:.WORD    0                  ; Word to receive channel number
                    223E                                     ;  assigned to the mailbox.
                    223E
 00000000 00000000  223E IOSB:    .QUAD   0                  ; Reserve one quadword for I/O status
                    2246                                     ;  block.
                    2246
                00  2246 IN_DEV: .BYTE    0                  ; Input device type.
                00  2247 OUT_DEV:.BYTE    0                  ; Output device type.
                    2248
                    2248 LINE_NUM:                           ; Current line number
          00000001  2248          .LONG   1                  ;
                    224C COL_NUM:                            ; Current column number
          00000001  224C          .LONG   1                  ;
                    2250 FIRST_LINE:                         ; First line on the screen.
          00000001  2250          .LONG   1                  ;
                    2254 FIRST_DATA_LINE:                    ; Number of the first line to contain
          00000007  2254          .LONG   7                  ; display data which can be vertically
                    2258                                     ;  scrolled.
                    2258 LAST_DATA_LINE:                     ; Number of the last line used to
          00000000  2258          .LONG   0                  ;  display data. (3rd to last on line
                    225C                                     ;  on the terminal)
```

```
                            225C  CMD_LINE:                                 ; Number of the second to last line on
                  00000000  225C          .LONG   0                         ;  the terminal.
                            2260  LAST_LINE:                                ; Number of the last line on the
                  00000000  2260          .LONG   0                         ;  terminal.
                            2264
                            2264
                            2264  ;
                            2264  ; Parameters used by the FIND_ENTRY and LOOKUP_FIELD routines.  These
                            2264  ; parameters control which field descriptors to use to display the report.
                            2264  ;
                            2264
                            2264  ADRS_TABLE:                               ; Address of table of field descriptor
                  00000000  2264          .LONG   0                         ;  addresses.
                            2268  BITMAP_ADRS:                              ; Address of bitmap to be scanned.
                  00000000  2268          .LONG   0
                            226C  MAP_LEN:                                  ; Length of the bitmap to be scanned.
                  00000000  226C          .LONG   0
                            2270  START_BIT:                                ; Number of the bit to start the scan.
                  00000000  2270          .LONG   0
                            2274  KEYTAB_ADRS:                              ; Address of table of keywords used to
                  00000000  2274          .LONG   0                         ;  identify a field
                            2278
                            2278
                            2278  ;
                            2278  ; List of systems that have been removed from the display by system ID number.
                            2278  ;
                            2278
                            2278  SYS_ID_LIST:
                  000022D8  2278          .BLKB   6*MAX_ID                  ; Reserve six bytes for each possible
                  00000060  22D8  ID_LIST_LEN = .-SYS_ID_LIST
                            22D8                                            ;  system.
                  00000010  22D8  ID_LIST_ENTRYS  =       MAX_ID            ; Maximum number of entries in the list.
                            22D8
                            22D8  ID_COUNT:                                 ; Number of entries in the list.
                  00000000  22D8          .LONG   0
                            22DC
                            22DC  ID_RESULT:                                ; Reserve one quadword for result from
         00000000 00000000  22DC          .QUAD   0                         ;  system ID string to hex conversion.
                            22E4
                            22E4
```

```
22E4            .SBTTL  Macro definitions.
22E4            .SBTTL          -       Conditional Branch Macro.
22E4
22E4    ;
22E4    ; The following macros are used in place of a BRANCH ON (CONDITION) instruction
22E4    ; when the displacement exceeds 256 bytes.
22E4    ;
22E4
22E4    ;
22E4    ; JLBS - used in place of BLBS
22E4    ;
22E4
22E4            .MACRO  JLBS    SOURCE,ADDRESS?L1
22E4                    BLBC    SOURCE,L1
22E4                    BRW     ADDRESS
22E4    L1:
22E4            .ENDM   JLBS
22E4
22E4
22E4    ;
22E4    ; JLBC - used in place of BLBC
22E4    ;
22E4
22E4            .MACRO  JLBC    SOURCE,ADDRESS?L1
22E4                    BLBS    SOURCE,L1
22E4                    BRW     ADDRESS
22E4    L1:
22E4            .ENDM   JLBC
22E4
22E4
22E4    ;
22E4    ; JNEQ - used in place of BNEQ
22E4    ;
22E4
22E4            .MACRO  JNEQ    ADDRESS?L1
22E4                    BEQL    L1
22E4                    BRW     ADDRESS
22E4    L1:
22E4            .ENDM   JNEQ
22E4
22E4
22E4    ;
22E4    ; JEQL - used in place of BEQL
22E4    ;
22E4
22E4            .MACRO  JEQL    ADDRESS?L1
22E4                    BNEQ    L1
22E4                    BRW     ADDRESS
22E4    L1:
22E4            .ENDM   JEQL
22E4
22E4
22E4    ;
22E4    ; JNEQU - used in place of BNEQU
22E4    ;
22E4
22E4            .MACRO  JNEQU   ADDRESS?L1
```

```
22E4                      BEQLU   L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JNEQU
22E4
22E4
22E4      ;
22E4      ; JEQLU - used in place of BEQLU
22E4      ;
22E4
22E4              .MACRO  JEQLU   ADDRESS?L1
22E4                      BNEQU   L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JEQLU
22E4
22E4
22E4      ;
22E4      ; JGTR - used in place of BGTR
22E4      ;
22E4
22E4              .MACRO  JGTR    ADDRESS?L1
22E4                      BLEQ    L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JGTR
22E4
22E4
22E4      ;
22E4      ; JLEQ - Used In place of BLEQ
22E4      ;
22E4
22E4              .MACRO  JLEQ    ADDRESS?L1
22E4                      BGTR    L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JLEQ
22E4
22E4
22E4      ;
22E4      ; JGTRU - used in place of BGTRU
22E4      ;
22E4
22E4              .MACRO  JGTRU   ADDRESS?L1
22E4                      BLEQU   L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JGTRU
22E4
22E4
22E4      ;
22E4      ; JLEQU - used in place of BLEQU
22E4      ;
22E4
22E4              .MACRO  JLEQU   ADDRESS?L1
22E4                      BGTRU   L1
22E4                      BRW     ADDRESS
```

```
22E4  L1:
22E4          .ENDM   JLEQU
22E4
22E4
22E4  ;
22E4  ; JGEQ - used in place of BGEQ
22E4  ;
22E4
22E4          .MACRO  JGEQ    ADDRESS?L1
22E4                  BLSS    L1
22E4                  BRW     ADDRESS
22E4  L1:
22E4          .ENDM   JGEQ
22E4
22E4
22E4  ;
22E4  ; JLSS - used in place of BLSS
22E4  ;
22E4
22E4          .MACRO  JLSS    ADDRESS?L1
22E4                  BGEQ    L1
22E4                  BRW     ADDRESS
22E4  L1:
22E4          .ENDM   JLSS
22E4
22E4
22E4  ;
22E4  ; JGEQU - used in place of BGEQU
22E4  ;
22E4
22E4          .MACRO  JGEQU   ADDRESS?L1
22E4                  BLSSU   L1
22E4                  BRW     ADDRESS
22E4  L1:
22E4          .ENDM   JGEQU
22E4
22E4
22E4  ;
22E4  ; JLSSU - used in place of BLSSU
22E4  ;
22E4
22E4          .MACRO  JLSSU   ADDRESS?L1
22E4                  BGEQU   L1
22E4                  BRW     ADDRESS
22E4  L1:
22E4          .ENDM   JLSSU
22E4
22E4
22E4  ;
22E4  ; JCS - used in place of BCS
22E4  ;
22E4
22E4          .MACRO  JCS     ADDRESS?L1
22E4                  BCC     L1
22E4                  BRW     ADDRESS
22E4  L1:
22E4          .ENDM   JCS
```

```
22E4
22E4
22E4      ;
22E4      ; JCC - used in place of BCC
22E4      ;
22E4
22E4              .MACRO  JCC     ADDRESS?L1
22E4                      BCS     L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JCC
22E4
22E4
22E4      ;
22E4      ; JVS - used in place of BVS
22E4      ;
22E4
22E4              .MACRO  JVS     ADDRESS?L1
22E4                      BVC     L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JVS
22E4
22E4
22E4      ;
22E4      ; JVC - used in place of BVC
22E4      ;
22E4
22E4              .MACRO  JVC     ADDRESS?L1
22E4                      BVS     L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JVC
22E4
22E4      ;
22E4      ; JBC - used in place of BBC
22E4      ;
22E4
22E4              .MACRO  JBC     BIT_NUM,SRC_ADRS,ADDRESS?L1
22E4                      BBS     BIT_NUM,SRC_ADRS,L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JBC
22E4
22E4      ;
22E4      ; JBS - used in place of BBS
22E4      ;
22E4
22E4              .MACRO  JBS     BIT_NUM,SRC_ADRS,ADDRESS?L1
22E4                      BBC     BIT_NUM,SRC_ADRS,L1
22E4                      BRW     ADDRESS
22E4      L1:
22E4              .ENDM   JBS
22E4
22E4
```

```
22E4            .SBTTL          -       DEFINE_FIELD macro.
22E4    ;
22E4    ; MACRO DEFINE_FIELD
22E4    ;
22E4    ;       This macro builts the data structures neccessary to display each field.
22E4    ;
22E4    ;
22E4    ; ARGUMENTS -
22E4    ;
22E4    ;       REPORT -
22E4    ;               Report that the field will appear in.  Must be CLUSTER,
22E4    ;               or LOCAL_PORTS.
22E4    ;
22E4    ;       OFFSET -
22E4    ;               Name of the SCS offset to use when extracting the data from the
22E4    ;               SCS database.  This can be any predefined offset.  It is
22E4    ;               normally one of the SCS offsets with prefixes CLUB$, SB$, CSB$,
22E4    ;               PB$, CDT$, or PDT$ or a miscellaneous buffer offset with prefix
22E4    ;               MB$.  This allows the default display routine DISPLAY_FIELD to
22E4    ;               be used.
22E4    ;
22E4    ;       LENGTH -
22E4    ;               The number of bytes of data to be extracted from the SCS
22E4    ;               database to display the field.
22E4    ;
22E4    ;               Note: For ASCIC formated fields, LENGTH must be the maximum
22E4    ;               number of characters in the length of the formatted string.
22E4    ;               This may differ from the number of bytes extracted if the
22E4    ;               DISPLAY_FIELD routine is not used for an ASCIC field.
22E4    ;
22E4    ;       TITLE -
22E4    ;               Title for this field as it will appear in the display.
22E4    ;
22E4    ;       FORMAT -
22E4    ;               Format of display for this field.  Must be one of HEX, DEC,
22E4    ;               ASCII, ASCIC, or TIME.
22E4    ;
22E4    ;       MINIMUM_SIZE -
22E4    ;               Minimum number of columns to allow for display of this field.
22E4    ;
22E4    ;       DEFAULT_SIZE -
22E4    ;               Default number of columns to use for display of this field.
22E4    ;
22E4    ;       DISPLAY_FLAG -
22E4    ;               This is an optional argument.  It is non-zero if the display of
22E4    ;               the field is to be enabled in the report by default.  If this
22E4    ;               argument is omitted, the field will not be displayed.
22E4    ;
22E4    ;       DISPLAY_ROUTINE -
22E4    ;               Optional address of routine to handle display of this field.
22E4    ;               If this argument is omitted, the field is displayed using the
22E4    ;               routine DISPLAY_FIELD.
22E4    ;
22E4    ;
22E4    ;       Five structures are built for each report:
22E4    ;
22E4    ;       'REPORT'_DSCRS -
```

```
22E4 ;                          This structure contains the field descriptors which contain the
22E4 ;                          information needed to display each of the fields in the report.
22E4 ;                          The following illustrates the format of one such descriptor:
22E4 ;
22E4 ; 'OFFSET'_DSCR:
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;                                  !  MAX_  !  MAX_  !  MIN_  ! FIELD_ !
22E4 ;                                  ! FIELD_ ! DATA_  ! FIELD_ !  SIZE  !
22E4 ;                                  !  SIZE  !  SIZE  !  SIZE  !        !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;                                  !    RESERVED     ! FORMAT ! LENGTH !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;                                  !        FD$L_DISPLAY_ROUTINE       !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;                                  !            FD$L_CLASS             !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;                                  !            FD$L_OFFSET            !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;                                  !          FD$L_OFFSET_PRFX         !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;
22E4 ; 'TITLE'_ADRS:
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;                                  !                          !  SIZE  !
22E4 ;                                  !                          +--------+
22E4 ;                                  -            FD$T_TITLE             -
22E4 ;                                  !                                   !
22E4 ;                                  !                                   !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ;
22E4 ;
22E4 ;
22E4 ;              'REPORT'_TABLE -
22E4 ;
22E4 ;                          Since the field descriptors are of variable length, this table
22E4 ;                          is used to locate the start of each descriptor.  It contains a
22E4 ;                          list of pointers to each of the field descriptors for the
22E4 ;                          report.
22E4 ;
22E4 ;              ^                   +--------+--------+--------+--------+
22E4 ;              !                   !    ADDRESS OF FIELD DESCRIPTOR    !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ; Number of fields                                   .
22E4 ;                                                    .
22E4 ;              !                                     .
22E4 ;              v                                     .
22E4 ;
22E4 ;
22E4 ;              'REPORT'_BITMAP -
22E4 ;
22E4 ;                          This bitmap is used to control which field in the report are
22E4 ;                          displayed.  Each bit in the map corresponds to one of the
22E4 ;                          fields in the display.  Setting the bit turns the field on.
22E4 ;
22E4 ;              ^                   +--------+--------+--------+--------+
22E4 ;              !                   ! ONE BIT FOR EACH FIELD IN REPORT  !
22E4 ;                                  +--------+--------+--------+--------+
22E4 ; Number of fields/32                                .
```

```
          22E4  ;                                                   .
          22E4  ;          :                                        .
          22E4  ;          v                                        .
          22E4  ;
          22E4  ;
          22E4  ;          'REPORT'_KEYTAB -
          22E4  ;
          22E4  ;                  This is a table of keywords which is used to associate a field
          22E4  ;                  title with a bit in the bitmap.  The format of this table is
          22E4  ;                  compatable with LIB$LOOKUP_KEY.
          22E4  ;
          22E4  ;                          +--------+--------+--------+--------+
          22E4  ;                          :            VECTOR_COUNT           :
          22E4  ;                          +--------+--------+--------+--------:
          22E4  ;          ^               +--------+--------+--------+--------+
          22E4  ;          :               :            KEYWORD VALUE          :
          22E4  ;                          +--------+--------+--------+--------:
          22E4  ;  Number of fields        :  ADDRESS OF KEYWORD NAME STRING   :
          22E4  ;    times two             +--------+--------+--------+--------:
          22E4  ;                                           .
          22E4  ;          :                                .
          22E4  ;          v                                .
          22E4  ;
          22E4  ;          'REPORT'_SIZE_ARRAY -
          22E4  ;
          22E4  ;                  This array is used to restore the default widths for each of the
          22E4  ;                  fields in the display.  It contains one byte for each of the
          22E4  ;                  fields in the display.  Each byte is a copy of the byte at
          22E4  ;                  FD$B_FIELD_SIZE in the field descriptor.
          22E4  ;
          22E4  ;          ^               +--------+
          22E4  ;          :               :Copy of :
          22E4  ;          :               :FIELD_  :
          22E4  ;                          : SIZE   :
          22E4  ; One byte for each        +--------+
          22E4  ;     field                    .
          22E4  ;                              .
          22E4  ;          :                   .
          22E4  ;          v
          22E4  ;
          22E4  ;
          22E4
00000000  22E4  KEYWRDVAL       =       0
          22E4
          22E4          .NOSHOW MEB
          22E4
          22E4          .MACRO  DEFINE_FIELD    REPORT,OFFSET,LENGTH,TITLE,FORMAT-
          22E4                                  MINIMUM_SIZE,DEFAULT_SIZE,DISPLAY_FLAG=0,-
          22E4                                  DISPLAY_ROUTINE=DISPLAY_FIELD
          22E4
          22E4
          22E4          .PSECT  'REPORT'_SIZE_ARRAY,RD,WRT,NOEXE
          22E4
          22E4          .BYTE   DEFAULT_SIZE            ; Save the default field size for use
          22E4                                          ;  by the INITIALIZE command.
          22E4
          22E4          .PSECT  'REPORT'_DSCRS,RD,WRT,NOEXE
```

```
22E4
22E4    'OFFSET'_DSCR:
22E4            .BYTE   DEFAULT_SIZE                    ; Store the current display field size
22E4                                                    ;  at offset FD$B_FIELD_SIZE.
22E4            .BYTE   MINIMUM_SIZE                    ; Store the minimum display field size
22E4                                                    ; at offset FD$B_MIN_FLD_SIZE.
22E4
22E4    ;
22E4    ; Store the maximum data size at offset FD$B_MAX_DATA_SIZE followed by maximum
22E4    ; field size at offset FD$B_MAX_FLD_SIZE if data is to be displayed using ASCII
22E4    ; or hexadecimal format.
22E4    ;
22E4
22E4            .IF     NE      FORMAT-DEC
22E4            .IF     NE      FORMAT-ASCIC
22E4            .IF     NE      FORMAT-TIME
22E4            .BYTE   FORMAT*LENGTH
22E4            .IF     GE      FORMAT*LENGTH-%LENGTH(TITLE)
22E4            .BYTE   FORMAT*LENGTH
22E4            .IFF
22E4            .BYTE   %LENGTH(TITLE)
22E4            .ENDC
22E4            .ENDC
22E4            .ENDC
22E4            .ENDC
22E4
22E4    ;
22E4    ; Store the maximum data size at offset FD$B_MAX_DATA_SIZE followed by maximum
22E4    ; field size at offset FD$B_MAX_FLD_SIZE if data is to be displayed using ASCIC
22E4    ; format.
22E4    ;
22E4            .IF     EQ      FORMAT-ASCIC
22E4            .BYTE   FORMAT/ASCIC*LENGTH
22E4            .IF     GE      FORMAT/ASCIC*LENGTH-%LENGTH(TITLE)
22E4            .BYTE   FORMAT/ASCIC*LENGTH
22E4            .IFF
22E4            .BYTE   %LENGTH(TITLE)
22E4            .ENDC
22E4            .ENDC
22E4
22E4    ;
22E4    ; Store the maximum data size at offset FD$B_MAX_DATA_SIZE followed by maximum
22E4    ; field size at offset FD$B_MAX_FLD_SIZE if data is to be displayed as a time
22E4    ; string.
22E4    ;
22E4            .IF     EQ      FORMAT-TIME
22E4            .BYTE   LENGTH
22E4            .IF     GE      LENGTH-%LENGTH(TITLE)
22E4            .BYTE   LENGTH
22E4            .IFF
22E4            .BYTE   %LENGTH(TITLE)
22E4            .ENDC
22E4            .ENDC
22E4
22E4    ;
22E4    ; Store the maximum data size at offset FD$B_MAX_DATA_SIZE followed by maximum
22E4    ; field size at offset FD$B_MAX_FLD_SIZE if data is to be displayed using
```

```
22E4   ; decimal format.
22E4   ;
22E4            .IF     EQ      FORMAT-DEC
22E4            LEN = 1
22E4            .IRP    SIZE,<3,5,8,10,13,15,17,20>
22E4            .IIF    EQ      LENGTH-LEN,     DEC_SIZE        =       SIZE
22E4            LEN = LEN +1
22E4            .ENDR
22E4            .BYTE   DEC_SIZE
22E4            .IF     GE      DEC_SIZE-%LENGTH(TITLE)
22E4            .BYTE   DEC_SIZE
22E4            .ENDC
22E4            .IF     LT      DEC_SIZE-%LENGTH(TITLE)
22E4            .BYTE   %LENGTH(TITLE)
22E4            .ENDC
22E4            .ENDC
22E4
22E4            .BYTE   LENGTH                          ; Store the number of bytes that appear
22E4                                                    ;  in the SCS database for this field
22E4                                                    ;  at offset FD$B_LENGTH.
22E4            .BYTE   FORMAT                          ; Store the type of format to use to
22E4                                                    ;  display the data at offset
22E4                                                    ;  FD$B_FORMAT.
22E4            .WORD   0                               ; Reserve on one.
22E4            .ADDRESS -                              ; Store the address of the routine used
22E4                    DISPLAY_ROUTINE                 ;  to display the field at offset
22E4                                                    ; FD$L_DISPLAY_ROUTINE.
22E4            .LONG   CLASS                           ; Store the family type that this field
22E4                                                    ;  is a member of at offset FD$L_CLASS.
22E4            .LONG   'OFFSET'                        ; Store the value of the SCS data
22E4                                                    ;  structure offset for this field at
22E4                                                    ; offset FD$L_OFFSET.
22E4   ;
22E4   ; Save the first four characters of the offset name at offset
22E4   ; FD$L_OFFSET_PRFX.
22E4   ;
22E4            .IIF EQ %LOCATE(<CLUB$>,OFFSET),        .LONG   ^A/CLUB/
22E4            .IIF EQ %LOCATE(<SB$>,OFFSET),          .LONG   ^A/SB$ /
22E4            .IIF EQ %LOCATE(<CSB$>,OFFSET),         .LONG   ^A/CSB$/
22E4            .IIF EQ %LOCATE(<PB$>,OFFSET),          .LONG   ^A/PB$ /
22E4            .IIF EQ %LOCATE(<CDT$>,OFFSET),         .LONG   ^A/CDT$/
22E4            .IIF EQ %LOCATE(<PDT$>,OFFSET),         .LONG   ^A/PDT$/
22E4            .IIF EQ %LOCATE(<MB$>,OFFSET),          .LONG   ^A/MB$ /
22E4            .IIF EQ %LOCATE(<SCS$>,OFFSET),         .LONG   ^A/SCS$/
22E4
22E4            .IIF EQ %LOCATE(<$>,OFFSET) - %LENGTH(OFFSET),  .LONG   0
22E4
22E4   ;
22E4   ; Store the ascii title at offset FD$T_TITLE.  Create a label for the
22E4   ; location where the title is stored.  The label is the same as the title, but
22E4   ; is followed by '_ADRS'.
22E4   ;
22E4
22E4   'TITLE'_ADRS:
22E4
22E4
22E4   ;
```

```
22E4    ; Now save the title as an ASCIC string.  This will place the length of the
22E4    ; string at offset FD$B_TITLE_SIZE and the string at offset FD$T_TITLE.
22E4    ;
22E4            .ASCIC  /TITLE/                         ; Title for display field.
22E4                                                    ; FD$B_TITLE_SIZE
22E4                                                    ; FD$T_TITLE
22E4
22E4            .PSECT  'REPORT'_TABLE,RD,NOWRT,NOEXE
22E4
22E4            .ADDRESS -                              ; Store descriptor address in the
22E4                    'OFFSET'_DSCR                   ;  address table.
22E4    'REPORT'_TAB_LEN = .-'REPORT'_TABLE             ; Calculate length of the table.
22E4
22E4            .PSECT  'REPORT'_BITMAP,RD,WRT,NOEXE
22E4
22E4    ;
22E4    ; Every 33 times we invoke this macro, save the mask we are building in the
22E4    ; bitmap and reset the mask to 0.  Also increment the count of the number of
22E4    ; longwords in the map.
22E4    ;
22E4
22E4    COUNT           =       COUNT+1
22E4            .IF     EQ      COUNT-33
22E4            .LONG   MASK                            ; Save the mask in the bitmap.
22E4
22E4    MAP_LENGTH      =       MAP_LENGTH+1            ; Increase count of the number of
22E4                                                    ;  longwords in the map.
22E4    MASK            =       0                       ; Clear the mask.
22E4    COUNT           =       1                       ; Reinitialize the counter.
22E4    MASK_BIT        =       1                       ; Start over with bit 0.
22E4
22E4            .ENDC
22E4
22E4    ;
22E4    ; If the display flag for this field descriptor is non-zero the this field is
22E4    ; to be displayed.  Set the bit for this field by combining the bit value for
22E4    ; this field with the bit mask.
22E4    ;
22E4
22E4            .IF     NE      DISPLAY_FLAG
22E4    MASK            =       MASK!MASK_BIT
22E4            .ENDC
22E4
22E4    ;
22E4    ; On to the next bit.
22E4    ;
22E4
22E4    MASK_BIT        =       MASK_BIT@1
22E4
22E4
22E4            .PSECT  'REPORT'_KEYTABLE
22E4
22E4
22E4    ;
22E4    ; Save the address of the title string in the keyword table.  This address is
22E4    ; used by LIB%LOOKUP_KEY.
22E4    ;
```

```
22E4
22E4              .ADDRESS -
22E4                      'TITLE'_ADRS
22E4
22E4
22E4    ;
22E4    ; Save the keyword value in the keyword table.  This value is used by
22E4    ; LIB$LOOKUP_KEY.
22E4    ;
22E4
22E4              .LONG   KEYWRDVAL
22E4
22E4    KEYWRDVAL         =       KEYWRDVAL+1
22E4
22E4              .ENDM   DEFINE_FIELD
22E4
22E4
```

```
              22E4   2460              .SBTTL  Cluster report field descriptors.
              22E4   2461              .SAVE_PSECT
              22E4   2462
              22E4   2463 ;
              22E4   2464 ; Initialize counters and masks.
              22E4   2465 ;
              22E4   2466
    00000001  22E4   2467 MASK_BIT          =       1
    00000000  22E4   2468 MASK              =       0
    00000000  22E4   2469 COUNT             =       0
    FFFFFFFF  22E4   2470 MAP_LENGTH        =       -1
              22E4   2471
              22E4   2472 ;
              22E4   2473 ; Create labels which will be used to reference each of the data structures
              22E4   2474 ; used in the cluster report as created by the DEFINE_FIELD macro.
              22E4   2475 ;
          00000000   2476              .PSECT  CLUSTER_SIZE_ARRAY,RD,WRT,NOEXE
              0000   2477 CLUSTER_SIZE_ARRAY:
              0000   2478
          00000000   2479              .PSECT  CLUSTER_BITMAP,RD,WRT,NOEXE
              0000   2480 CLUSTER_BITMAP:
              0000   2481
          00000000   2482              .PSECT  CLUSTER_TABLE,RD,NOWRT,NOEXE
              0000   2483 CLUSTER_TABLE:
              0000   2484
          00000000   2485              .PSECT  CLUSTER_KEYTABLE,RD,NOWRT,NOEXE
              0000   2486 CLUSTER_KEYTAB:
    00000000  0000   2487              .LONG   0
              0004   2488
              0004   2489 ;
              0004   2490 ; Define the fields that make up the CLUSTER class.
              0004   2491 ;
              0004   2492
    00000001  0004   2493 CLASS    =         CLS$M_CLUB
              0004   2494
              0004   2495              DEFINE_FIELD    <CLUSTER>,<CLUB$W_QUORUM>,2,<CL_QUORUM>,DEC,4,4,1
              000C   2496              DEFINE_FIELD    <CLUSTER>,<CLUB$W_VOTES>,2,<CL_VOTES>,DEC,4,4,1
              0014   2497              DEFINE_FIELD    <CLUSTER>,<CLUB$T_QDNAME>,16,<QD_NAME>,ASCIC,6,15,1
              001C   2498              DEFINE_FIELD    <CLUSTER>,<CLUB$L_FLAGS>,Y[illegible]NO_STRNG_LEN,-
              001C   2499                              <QF_VOTE>,ASCIC,4,4,1,DISPLAY_QF_VOTE
              0024   2500              DEFINE_FIELD    <CLUSTER>,<CLUB$W_QDVOTES>,2,<CL_QDVOTES>,DEC,4,6
              002C   2501              DEFINE_FIELD    <CLUSTER>,<CLUB$W_MEMSEQ>,2,<MEMSEQ>,DEC,4,6
              0034   2502              DEFINE_FIELD    <CLUSTER>,<CLUB$Q_FTIME>,TIME_STRING_LEN,<FORMED>,-
              0034   2503                              TIME,6,15,1
              003C   2504              DEFINE_FIELD    <CLUSTER>,<CLUB$Q_LST_TIME>,TIME_STRING_LEN,-
              003C   2505                              <LAST_TRANSITION>,TIME,6,15,1
              0044   2506
              0044   2507 ;
              0044   2508 ; Define the fields that make up the SYSTEMS class.
              0044   2509 ;
              0044   2510
    00000002  0044   2511 CLASS    =         CLS$M_SYS
              0044   2512
              0044   2513              DEFINE_FIELD    <CLUSTER>,<SB$T_NODENAME>,16,<NODE>,ASCIC,2,6,1
              004C   2514              DEFINE_FIELD    <CLUSTER>,<SB$B_SYSTEMID>,6,<SYS_ID>,DEC,2,8
              0054   2515              DEFINE_FIELD    <CLUSTER>,<SB$B_HWVERS>,12,<HW_VERS>,HEX,2,20
    00000034  005C   2516 SB$T_HWTYPE = SB$Q_SWINCARN + 8
```

```
           005C        DEFINE_FIELD    <CLUSTER>,<SB$T_HWTYPE>,4,<HW_TYPE>,ASCII,4,4
           0064        DEFINE_FIELD    <CLUSTER>,<SB$T_SWVERS>,4,<SOFTWARE>,ASCII,3,8,1,-
           0064                        DISPLAY_SOFTWARE
           006C        DEFINE_FIELD    <CLUSTER>,<SB$SHC$L_NUM_PB>,2,<NUM_CIRCUITS>,DEC,7,7
           0074        DEFINE_FIELD    <CLUSTER>,<SB$W_MAXDG>,2,<DG_SIZE>,DEC,2,4
           007C        DEFINE_FIELD    <CLUSTER>,<SB$W_MAXMSG>,2,<MSG_SIZE>,DEC,2,5
           0084        DEFINE_FIELD    <CLUSTER>,<SB$Q_SWINCARN>,8,<INCARNATION>,HEX,2,14
0000002C   008C SB$Q_SWINCARN_2 = SB$Q_SWINCARN
           008C        DEFINE_FIELD    <CLUSTER>,<SB$Q_SWINCARN_2>,TIME_STRING_LEN,-
           008C                        <INCN_TIME>,TIME,6,9
           0094
           0094 ;
           0094 ; Define the fields that make up the MEMBERS class.
           0094 ;
           0094
00000004   0094 CLASS           =      CLS$M_CSB
           0094
           0094        DEFINE_FIELD    <CLUSTER>,<CSB$L_CSID>,4,<CSID>,HEX,5,6
           009C        DEFINE_FIELD    <CLUSTER>,<CSB$W_SENDSEQNM>,2,<SEND_SQ>,DEC,6,6
           00A4        DEFINE_FIELD    <CLUSTER>,<CSB$W_RCVDSEQNM>,2,<RCVD_SQ>,DEC,6,6
           00AC        DEFINE_FIELD    <CLUSTER>,<CSB$W_ACKRSEQNM>,2,<ACKR_SQ>,DEC,4,6
           00B4        DEFINE_FIELD    <CLUSTER>,<CSB$W_LCKDIRWT>,2,<DIR_WT>,DEC,4,6
           00BC        DEFINE_FIELD    <CLUSTER>,<CSB$B_UNACKEDMSGS>,1,<UNACKED>,DEC,3,5
           00C4        DEFINE_FIELD    <CLUSTER>,<CSB$B_REMACKLIM>,1,<ACK_LM>,DEC,4,5
           00CC        DEFINE_FIELD    <CLUSTER>,<CSB$B_ECOLVL>,2,<PROTOCOL>,DEC,3,5,,-
           00CC                        DISPLAY_PROTOCOL
           00D4        DEFINE_FIELD    <CLUSTER>,<CSB$B_WARMCDRPS>,1,<WARMCDRPS>,DEC,1,5
           00DC        DEFINE_FIELD    <CLUSTER>,<CSB$B_STATE>,CSB_STATE_STRNG_LEN,-
           00DC                        <CNX_STATE>,ASCIC,5,6,,DISPLAY_CSB_STATE
           00E4        DEFINE_FIELD    <CLUSTER>,<CSB$W_VOTES>,2,<VOTES>,DEC,1,5
           00EC        DEFINE_FIELD    <CLUSTER>,<CSB$W_QUORUM>,2,<QUORUM>,DEC,1,6
           00F4        DEFINE_FIELD    <CLUSTER>,<CSB$W_QDVOTES>,2,<QDVOTES>,DEC,4,7
           00FC
00000060   00FC CSB$L_STATUS1   =      CSB$L_STATUS
           00FC        DEFINE_FIELD    <CLUSTER>,<CSB$L_STATUS1>,YES_NO_STRNG_LEN,-
           00FC                        <QF_SAME>,ASCIC,4,7,,DISPLAY_QF_SAME
00000060   0104 CSB$L_STATUS2   =      CSB$L_STATUS
           0104        DEFINE_FIELD    <CLUSTER>,<CSB$L_STATUS2>,YES_NO_STRNG_LEN,-
           0104                        <QF_ACTIVE>,ASCIC,4,9,,DISPLAY_QF_ACTIVE
           010C        DEFINE_FIELD    <CLUSTER>,<CSB$L_STATUS>,CSB_STATUS_STRNG_LEN,-
           010C                        <STATUS>,ASCIC,3,7,1,DISPLAY_CSB_STATUS
           0114        DEFINE_FIELD    <CLUSTER>,<CSB$Q_REFTIME>,TIME_STRING_LEN,-
           0114                        <TRANSITION_TIME>,TIME,6,15,,
           011C ;
           011C ; Define the fields that make up the CIRCUITS class.
           011C ;
           011C
00000008   011C CLASS           =      CLS$M_CIR
           011C
           011C        DEFINE_FIELD    <CLUSTER>,<PB$T_LPORT_NAME>,4,<LPORT_NAME>,ASCII,3,5
           0124        DEFINE_FIELD    <CLUSTER>,<PB$B_RSTATION>,6,<RPORT_NUM>,DEC,2,5,1
           012C        DEFINE_FIELD    <CLUSTER>,<PB$B_RST_PORT>,1,<RP_OWNER>,DEC,5,6
           0134        DEFINE_FIELD    <CLUSTER>,<PB$L_RPORT_TYP>,RPORT_TYP_STRNG_LEN,-
           0134                        <RP_TYPE>,ASCIC,5,6,1,DISPLAY_RPORT_TYP
           013C        DEFINE_FIELD    <CLUSTER>,<PB$SHC$L_NUM_CDT>,4,<NUM_CONNECTIONS>,DEC,7,7
           0144        DEFINE_FIELD    <CLUSTER>,<PB$W_STATE>,PB_STATE_STRNG_LEN,-
           0144                        <CIR_STATE>,ASCIC,5,7,1,DISPLAY_CIR_STATE
```

```
         014C         DEFINE_FIELD    <CLUSTER>,<PB$B_RSTATE>,PB_STATE_STRNG_LEN,-
         014C                         <REM_STATE>,ASCIC,6,7,,DISPLAY_RSTATE
         0154         DEFINE_FIELD    <CLUSTER>,<PB$B_CBL_STS>,CB_STAT_STRNG_LEN,-
         0154                         <CABLE_STATUS>,ASCIC,5,5,,DISPLAY_CABLE_STATUS
         015C         DEFINE_FIELD    <CLUSTER>,<PB$L_RPORT_REV>,4,<RP_REVISION>,HEX,5,8
         0164         DEFINE_FIELD    <CLUSTER>,<PB$L_RPORT_FCN>,4,<RP_FUNCTIONS>,HEX,5,8
         016C         DEFINE_FIELD    <CLUSTER>,<PB$L_WAITQFL>,4,<SCS_WAITERS>,DEC,2,5,,
         0174
         0174    ;
         0174    ; Define the fields that make up the CONNECTIONS class.
         0174    ;
         0174
00000010 0174    CLASS          =     CLS$M_CON
         0174
         0174         DEFINE_FIELD    <CLUSTER>,<SCS$T_DST_PROC>,16,<LOC_PROC_NAME>,-
         0174                         ASCII,8,16,1,DISPLAY_PROC_NAME
         017C         DEFINE_FIELD    <CLUSTER>,<SCS$T_SRC_PROC>,16,<REM_PROC_NAME>,-
         017C                         ASCII,8,16,,DISPLAY_PROC_NAME
         0184         DEFINE_FIELD    <CLUSTER>,<CDT$W_STATE>,DISPLAY_CON_STATE_LEN,-
         0184                         <CON_STATE>,ASCIC,6,7,1,DISPLAY_CON_STATE
         018C         DEFINE_FIELD    <CLUSTER>,<CDT$L_LCONID>,4,<LOC_CONID>,HEX,8,9,
         0194         DEFINE_FIELD    <CLUSTER>,<CDT$L_RCONID>,4,<REM_CONID>,HEX,8,9,
         019C         DEFINE_FIELD    <CLUSTER>,<CDT$W_BLKSTATE>,DISPLAY_SCS_STATE_LEN,-
         019C                         <SCS_STATE>,ASCIC,6,7,,DISPLAY_SCS_STATE
         01A4
         01A4    ;
         01A4    ; Define the fields that make up the COUNTERS class.
         01A4    ;
         01A4
00000020 01A4    CLASS          =     CLS$M_CNT
         01A4         DEFINE_FIELD    <CLUSTER>,<CDT$L_DGSENT>,4,<DGS_SENT>,DEC,5,5,1,
         01AC         DEFINE_FIELD    <CLUSTER>,<CDT$L_DGRCVD>,4,<DGS_RCVD>,DEC,5,5,1,
         01B4         DEFINE_FIELD    <CLUSTER>,<CDT$L_DGDISCARD>,4,<DGS_DSCRD>,DEC,-
         01B4                         5,5,,
         01BC         DEFINE_FIELD    <CLUSTER>,<CDT$L_MSGSENT>,4,<MSGS_SENT>,DEC,6,6,1,
         01C4         DEFINE_FIELD    <CLUSTER>,<CDT$L_MSGRCVD>,4,<MSGS_RCVD>,DEC,6,6,1,
         01CC         DEFINE_FIELD    <CLUSTER>,<CDT$L_SNDDATS>,4,<BLKS_SENT>,DEC,6,6,,
         01D4         DEFINE_FIELD    <CLUSTER>,<CDT$L_REQDATS>,4,<BLKS_REQ>,DEC,3,6,,
         01DC         DEFINE_FIELD    <CLUSTER>,<CDT$L_BYTSENT>,4,<KB_SENT>,DEC,4,4,,-
         01DC                         DISPLAY_KBYTES
         01E4         DEFINE_FIELD    <CLUSTER>,<CDT$L_BYTREQD>,4,<KB_RCVD>,DEC,4,4,,-
         01E4                         DISPLAY_KBYTES
         01EC         DEFINE_FIELD    <CLUSTER>,<CDT$L_BYTMAPD>,4,<KB_MAPPED>,DEC,4,4,,-
         01EC                         DISPLAY_KBYTES
         01F4         DEFINE_FIELD    <CLUSTER>,<CDT$W_QCR_CNT>,2,<CR_WAITS>,DEC,1,4,,
         01FC         DEFINE_FIELD    <CLUSTER>,<CDT$W_QBDT_CNT>,2,<BDT_WAITS>,DEC,1,5,,
         0204
         0204    ;
         0204    ; Define the fields that make up the CREDITS class.
         0204    ;
         0204
00000040 0204    CLASS          =     CLS$M_CRD
         0204
         0204         DEFINE_FIELD    <CLUSTER>,<CDT$W_SEND>,2,<SEND>,-
         0204                         DEC,1,4,1,
         020C         DEFINE_FIELD    <CLUSTER>,<CDT$W_MINSEND>,2,<MIN_SEND>,-
         020C                         DEC,5,5,1,
```

```
              0214          DEFINE_FIELD    <CLUSTER>,<CDT$W_REC>,2,<RECEIVE>,-
              0214                          DEC,1,3,1,
              021C          DEFINE_FIELD    <CLUSTER>,<CDT$W_PENDREC>,2,<PEND_REC>,-
              021C                          DEC,1,6,1,
              0224          DEFINE_FIELD    <CLUSTER>,<CDT$W_INITLREC>,2,<INIT_REC>,-
              0224                          DEC,1,6,1,
              022C          DEFINE_FIELD    <CLUSTER>,<CDT$W_MINREC>,2,<MIN_REC>,-
              022C                          DEC,5,5,1,
              0234
          00000234          .PSECT  CLUSTER_KEYTABLE,RD,NOWRT,NOEXE
              0234
00000046      0234  NUM_CLUSTER_DSCRS = KEYWRDVAL                    ; Save the number of descriptors in the
              0234                                                   ;  cluster report.
              0234
00000000      0234  . =     CLUSTER_KEYTAB                           ; Save the vector count for
0000008C      0000          .LONG   KEYWRDVAL*2                      ;  LIB$LOOKUP_KEY.
00000094      0004  . =     .+KEYWRDVAL*2
              0094
              0094
              0094  ;
              0094  ; Save any partial mask in the bitmap and increase the count of the bitmap's
              0094  ; length.
              0094  ;
          000000C8          .PSECT  CLUSTER_BITMAP,RD,WRT,NOEXE
              0008
FFFFFFE5      0008          .IF     NE          COUNT-33
0000003F      0008          .LONG   MASK
00000002      000C          MAP_LENGTH      =           MAP_LENGTH+1
              000C          .ENDC
              000C
00000002      000C  CLUSTER_MAP_LEN = MAP_LENGTH                     ; Save the number of longwords in the
              000C                                                   ;  cluster bitmap.
              000C
              000C  CL_SAVE_BITMAP:                                  ; Reserve enough room for a copy of the
00000018      000C          .BLKL   MAP_LENGTH+1                     ;  cluster bitmap to be used by the
              0018                                                   ;  INITIALIZE command.
              0018
              0018
```

```
                0018          .SBTTL  Local status report field descriptors.
                0018  ;
                0018  ; Initialize counters and masks.
                0018  ;
                0018
      00000001  0018  MASK_BIT          =       1
      00000000  0018  MASK              =       0
      00000000  0018  COUNT             =       0
      FFFFFFFF  0018  MAP_LENGTH        =       -1
      00000000  0018  KEYWRDVAL         =       0
                0018
                0018  ;
                0018  ; Create labels which will be used to reference each of the data structures
                0018  ; used in the local report as created by the DEFINE_FIELD macro.
                0018  ;
                0018
          00000000    .PSECT  LOCAL_SIZE_ARRAY,RD,WRT,NOEXE
                0000  LOCAL_SIZE_ARRAY:
                0000
          00000000    .PSECT  LOCAL_BITMAP,RD,WRT,NOEXE
                0000  LOCAL_BITMAP:
                0000
          00000000    .PSECT  LOCAL_TABLE,RD,NOWRT,NOEXE
                0000  LOCAL_TABLE:
                0000
          00000000    .PSECT  LOCAL_KEYTABLE,RD,NOWRT,NOEXE
                0000  LOCAL_KEYTAB:
      00000000  0000          .LONG   0
                0004
                0004  ;
                0004  ; Define the fields that make up the LOCAL_PORTS class.
                0004  ;
                0004
      00000080  0004  CLASS             =       CLS$M_LPRT
                0004
                0004
                0004          DEFINE_FIELD    <LOCAL>,<LOCSYSID>,6,<LSYSTEM_ID>,DEC,4,4,,FORMAT_DATA
                000C          DEFINE_FIELD    <LOCAL>,<NUM_BUF_DSCR>,2,<BUFF_DESCRIPTORS>,-
                000C                          DEC,2,10,,FORMAT_DATA
                0014          DEFINE_FIELD    <LOCAL>,<NUM_FREE_BUF_DSCR>,4,-
                0014                          <FREE_BUFF_DESCRIPTORS>,DEC,2,10,,FORMAT_DATA
                001C
                001C
                001C          DEFINE_FIELD    <LOCAL>,<MB$L_PORT_NAME>,4,<NAME>,ASCII,3,4,1,
                0024          DEFINE_FIELD    <LOCAL>,<MB$L_STS>,PORT_STATUS_STRNG_LEN,-
                0024                          <LP_STATUS>,ASCIC,4,6,1,DISPLAY_PORT_STATUS
                002C          DEFINE_FIELD    <LOCAL>,<MB$L_PORT_NUM>,1,<PORT_NUM>,DEC,1,4,1
                0034          DEFINE_FIELD    <LOCAL>,<MB$W_DG_FREE>,2,<DGS_FREE>,ASCII,1,3,1
                003C          DEFINE_FIELD    <LOCAL>,<MB$W_MSG_FREE>,2,<MSGS_FREE>,ASCII,1,3,1
                0044          DEFINE_FIELD    <LOCAL>,<PDT$T_CNTOWNER>,16,<COUNTER_OWNER>,ASCII,8,16
                004C          DEFINE_FIELD    <LOCAL>,<MB$L_PL_WAITERS>,4,<POOL_WAITERS>,DEC,2,4
                0054          DEFINE_FIELD    <LOCAL>,<PDT$L_MSGHDRSZ>,4,<MSG_HDR_SIZE>,DEC,5,6
                005C          DEFINE_FIELD    <LOCAL>,<PDT$L_DGOVRHD>,4,<DG_OVRHD_SIZE>,DEC,4,6
                0064          DEFINE_FIELD    <LOCAL>,<MB$L_DEVTYPE>,DEV_TYP_STRNG_LEN,-
                0064                          <LP_TYPE>,ASCIC,4,5,,DISPLAY_PORT_TYPE
                006C          DEFINE_FIELD    <LOCAL>,<PDT$W_PBCOUNT>,2,<OPEN_CIRCS>,DEC,3,6,1
                0074          DEFINE_FIELD    <LOCAL>,<MB$L_FORM_PB>,4,<FORM_CIRCS>,DEC,3,6,1
```

```
                007C            DEFINE_FIELD    <LOCAL>,<MB$W_CMDS_QUED>,2,<CMDS_QUEUED>,ASCII,2,4
                0084            DEFINE_FIELD    <LOCAL>,<MB$W_RSPS_QUED>,2,<RSPS_QUEUED>,ASCII,2,4
                008C            DEFINE_FIELD    <LOCAL>,<PDT$B_MAX_PORT>,1,<MAX_PORT>,DEC,5,8
                0094            DEFINE_FIELD    <LOCAL>,<PDT$B_PORTMAP>,2,<PRT_MAP>,HEX,5,7
                009C            DEFINE_FIELD    <LOCAL>,<PDT$B_DQIMAP>,2,<DGI_MAP>,HEX,3,6
                00A4            DEFINE_FIELD    <LOCAL>,<PDT$B_PLOGMAP>,2,<LOG_MAP>,HEX,5,7
                00AC            DEFINE_FIELD    <LOCAL>,<PDT$B_PO_LBSTS>,LB_STAT_STRG_LEN,-
                00AC                            <LB_STATUS>,ASCIC,5,5,,DISPLAY_LOOPBACK_STATUS
                00B4
                00B4
                00B4    ;
                00B4    ; Define the fields that make up the ERRORS class.
                00B4    ;
                00B4
      00000100  00B4    CLASS           =       CLS$M_ERR
                00B4
                00B4
                00B4            DEFINE_FIELD    <LOCAL>,<MB$B_ERTCNT>,1,<ERT_COUNT>,DEC,3,5,1
                00BC            DEFINE_FIELD    <LOCAL>,<MB$B_ERTMAX>,1,<ERT_MAX>,DEC,5,5,1
                00C4            DEFINE_FIELD    <LOCAL>,<MB$W_ERRCNT>,1,<NUM_ERRORS>,DEC,6,6,1
                00CC
                00CC
        000000CC                .PSECT  LOCAL_KEYTABLE,RD,NOWRT,NOEXE
                00CC
                00CC
      00000019  00CC    NUM_LOCAL_DSCRS = KEYWRDVAL                     ; Save the number of descriptors in the
                00CC                                                    ;  local report.
                00CC
      00000000  00CC    . =     LOCAL_KEYTAB                            ; Save the vector count for
      00000032  0000            .LONG   KEYWRDVAL*2                     ;  LIB$LOOKUP_KEY.
      0000003A  0004    . =     .+KEYWRDVAL*2
                003A
        00000000                .PSECT  LOCAL_BITMAP,RD,WRT,NOEXE
                0000
                0000    ;
                0000    ; Save any partial mask in the bitmap and increase the count of the bitmap's
                0000    ; length.
                0000    ;
      FFFFFFF8  0000            .IF     NE      COUNT-33
      01C060F8  0000            .LONG   MASK
      00000000  0004            MAP_LENGTH      =       MAP_LENGTH+1
                0004            .ENDC
                0004
                0004
      00000000  0004    LOCAL_MAP_LEN = MAP_LENGTH                      ; Save the number of longwords in the
                0004                                                    ;  local bitmap.
                0004
                0004    LOC_SAVE_BITMAP:                                ; Reserve enough room for a copy of the
      00000008  0004            .BLKL   MAP_LENGTH+1                    ;  local bitmap to be used by the
                0008                                                    ;  INITIALIZE command.
                0008
                0008
        000022E4                .RESTORE_PSECT
                22E4
                22E4
```

```
                                22E4          .SBTTL  Main program loop.
                            00000000          .PSECT  MAIN,RD,NOWRT,EXE
                                0000  ;++
                                0000  ;
                                0000  ; FUNCTIONAL DESCRIPTION:
                                0000  ;
                                0000  ;       This is the main entry point for show cluster.  The basic program
                                0000  ;       flow is as follows:
                                0000  ;
                                0000  ;       1)      Initialize
                                0000  ;       2)      Capture a snapshot of the SCS database.
                                0000  ;       3)      Display the report.
                                0000  ;       4)      Set a timer and wait for expiration or unsolicited data.
                                0000  ;               (command input or broadcast message received)
                                0000  ;       5)      Process the unsolicited data and/or go to step 2.
                                0000  ;
                                0000  ;--
                                0000
                           0CBC 0000  .ENTRY  START,^M<R2,R3,R4,R5,R7,R10,R11>
                                0002
     00000476'EF    09     C8   0002          BISL    #<MF$M_INIT_FLAG!-            ; Set flags to indicate we're performing
                                0009                  MF$M_RIP>,MISC_FLAGS      ;  initialization and that the full
                                0009                                            ;  screen must be drawn.
            0856    30          0009          BSBW    INIT                      ; Initialize.
                                000C          JLBC    R0,MAIN_ERR               ; Branch on error.
                                0012
                                0012  MAIN_LOOP:
     00002247'EF    03     E0   0012          BBS     #DVT$V_UNKNOWN,OUT_DEV,-; Set up screen buffer on each pass if
                    08          0019                  10$                       ;  output device is an unknown terminal
                                001A                                            ;  type.
                    00     E1   001A          BBC     #MF$V_INIT_FLAG,-         ; Otherwise set up screen buffer only
  13 00000476'EF                001C                  MISC_FLAGS,20$            ;  during initialization.
     0000055B'EF           7F   0022  10$:    PUSHAQ  SCREEN_DSCR               ; Set up a buffer for the screen.
     00000000'GF    01     FB   0028          CALLS   #1,G^LIB$SET_BUFFER       ;
                                002F          JLBC    R0,MAIN_ERR               ; Branch on error.
                                0035
1F 00002247'EF      01     E1   0035  20$:    BBC     #DVT$V_DCRT,OUT_DEV,30$ ; Branch if output device isn't a
                                003D                                            ;  DECcrt.
     0000224C'EF           DF   003D          PUSHAL  COL_NUM                   ; Set line drawing and special graphics
     00002248'EF           DF   0043          PUSHAL  LINE_NUM                  ;  character set in G1 area in DECcrt.
     00000168'EF           DF   0049          PUSHAL  GRAPHICS_DSGNTR           ;
     00000000'GF    03     FB   004F          CALLS   #3,G^LIB$PUT_SCREEN       ;
                                0056          JLBC    R0,MAIN_ERR               ; Branch on error.
                                005C
            0F9A    30          005C  30$:    BSBW    DISPLAY                   ; Generate the display output.
                                005F          JLBC    R0,MAIN_ERR               ; Branch on error.
     00000000'GF    00     FB   0065          CALLS   #0,G^LIB$PUT_BUFFER       ; Output the report to the screen.
     00000476'EF    04     E0   006C          BBS     #MF$V_CONT,MISC_FLAGS,- ; Branch if continuous display.
                    22          0073                  40$                       ;
     0000224C'EF           DF   0074          PUSHAL  COL_NUM                   ; Otherwise place the cursor at the
     0000225C'EF           DF   007A          PUSHAL  CMD_LINE                  ;  bottom of the screen.
     00000000'GF    02     FB   0080          CALLS   #2,G^LIB$SET_CURSOR       ;
                                0087          JLBC    R0,MAIN_ERR               ; Branch on error.
                                008D          $EXIT_S                           ; Exit.
                                0096
                                0096  ;
                                0096  ; If input is from a command file, then continute reading commands from the
```

```
                                 0096  ; file.  Otherwise, go to sleep until a key in pressed, a broadcast message
                                 0096  ; comes in, or the interval timer expires.
                                 0096  ;
                                 0096
    39 00002247'EF     02    E1  0096  40$:    BBC     #DVT$V_HARD,OUT_DEV,50$ ; Branch if not hardcopy terminal.
           0000055B'EF     7F    009E          PUSHAQ  SCREEN_DSCR             ; Set up a buffer for the screen.
       00000000'GF     01    FB  00A4          CALLS   #1,G^LIB$SET_BUFFER     ;
                                 00AB          JLBC    R0,MAIN_ERR             ; Branch on error.
    1E 00002246'EF     05    E1  00B1          BBC     #DVT$V_DISK,IN_DEV,50$  ; Branch if input isn't from a file.
       00001215'EF     00    FB  00B9          CALLS   #0,EXECUTE              ; Go process command input.
                                 00C0          JLBC    R0,MAIN_ERR             ; Branch on error.
                       00    E1  00C6          BBC     #MF$V_INIT_FLAG,-       ; Branch if initialization not in
         09 00000476'EF          00C8                  MISC_FLAGS,50$          ;  progress.
                                 00CE          $EXIT_S                         ; Otherwise exit.
                                 00D7
                                 00D7  50$:    $SETIMR_S -                     ; Set the refresh interval timer.
                                 00D7                  EFN=#EVF$V_TIM_FLAG,-   ;
                                 00D7                  DAYTIM=DELAY,-          ;
                                 00D7                  REQIDT=#1               ;
                                 00EA          JLBC    R0,MAIN_ERR             ; Branch on error.
                                 00F0
                                 00F0  ;
                                 00F0  ; Wait for expiration of interval timer, or an AST to occur.
                                 00F0  ;
                                 00F0
                                 00F0          $WFLOR_S -
                                 00F0                  EFN=#0,-
                                 00F0                  MASK=#EVF$M_TIM_FLAG!EVF$M_AST_FLAG
                                 00FB          JLBC    R0,MAIN_ERR             ; Branch on error.
                                 0101
                                 0101          $CANTIM_S -                     ; Now that we're awake, cancel
                                 0101                  REQIDT=#1               ;  outstanding refresh timer requests.
                                 010C          JLBC    R0,MAIN_ERR             ; Branch on error.
                                 0112
                                 0112          $READEF_S -                     ; Read flags to determine which
                                 0112                  EFN=#0,-                ;  condition occured.
                                 0112                  STATE=FLAG_STATE
                                 0121          JLBC    R0,MAIN_ERR             ; Branch on error.
                                 0127
                       02    E0  0127          BBS     #EVF$V_AST_FLAG,-       ; Branch if AST flag set.
       21 00000472'EF            0129                  FLAG_STATE,70$
       00000476'EF     05    E0  012F          BBS     #MF$V_CYCLE,MISC_FLAGS,-; Branch if we're not cycling the
                       03        0136                  60$                     ;   display.
                     FED8    31  0137          BRW     MAIN_LOOP               ; Otherwise start over.
           000021AB'EF     96    013A  60$:    INCB    SWITCH                  ; Toggle switch.
                                 0140          JLBS    SWITCH,MAIN_LOOP        ; Branch if not time to switch displays.
                     2419    30  014A          BSBW    SWITCH_DISPLAY          ; Otherwise, switch the displays
                     FEC2    31  014D          BRW     MAIN_LOOP               ;  and do a screen update.
                                 0150  70$:    $CLREF_S -                      ; Reset AST flag.
                                 0150                  EFN=#EVF$V_AST_FLAG     ;
                                 0159          JLBC    R0,MAIN_ERR             ; Branch on error.
        01     00001FB5'EF   B1  015F          CMPW    MBX_BUF,#MSG$_TRMUNSOLIC; Was AST caused by unsolicited data?
                       0E    12  0166          BNEQ    80$                     ; Branch if not.
           00000199'EF     16    0168          JSB     READ_LOOP               ; Otherwise go process the command.
                                 016E          JLBC    R0,MAIN_ERR             ; Branch on error.
                       17    11  0174          BRB     MAIN_CONT
  0053 8F      00001FB5'EF   B1  0176  80$:    CMPW    MBX_BUF,#MSG$_TRMBRDCST ; Was AST caused by terminal broadcast?
```

```
          0C  12 017F  2897          BNEQ    MAIN_CONT                ; If not, ignore AST.
   0000045B'EF  16 0181  2898          JSB     BRDCST                   ; Otherwise, go handle broadcast
                 0187  2899                                           ;   message.
                 0187  2900          JLBC    R0,MAIN_ERR              ; Branch on error.
                 018D  2901
                 018D  2902 MAIN_CONT:
        18A9  30 018D  2903          BSBW    SET_AST                  ; Set up for next AST.
                 0190  2904          JLBC    R0,MAIN_ERR              ; Branch on error.
        FE79  31 0196  2905          BRW     MAIN_LOOP                ; Start again.
                 0199  2906
                 0199  2907          .DSABL  LSB
                 0199  2908
```

```
                                0199  ;++
                                0199  ;
                                0199  ; FUNCTIONAL DESCRIPTION:
                                0199  ;
                                0199  ;       This routine is entered when a unsolicited data AST occurs.  It
                                0199  ;       examines the type-ahead buffer and then issues a read QIO with or
                                0199  ;       without a prompt depending on the character found in the type-ahead
                                0199  ;       buffer.
                                0199  ;
                                0199  ;       Once the read completes control is passed to DCL_PARSE or to
                                0199  ;       the appropriate routine to handle control or escape sequences.
                                0199  ;
                                0199  ; CALLING SEQUENCE:
                                0199  ;
                                0199  ;       JSB     READ_LOOP
                                0199  ;
                                0199  ; CALLED BY:
                                0199  ;
                                0199  ;       MAIN_LOOP
                                0199  ;
                                0199  ;--
                                0199
                                0199          .ENABL  LSB
                                0199
                                0199  READ_LOOP:
                                0199          $QIOW_S -                       ; Get the number of characters in the
                                0199                  CHAN=IN_CHAN,-          ;  type ahead buffer.
                                0199                  FUNC=#IO$_SENSEMODE!IO$M_TYPEAHDCNT,-
                                0199                  P1=TYPEAHD_BUF
                                01BE          JLBC    R0,999$                 ; Branch on error.
           000004BD'EF   B5     01C4          TSTW    TYPEAHD_BUF             ; Are there any characters in the type
                                01CA                                          ;  ahead buffer?
                    03   12     01CA          BNEQU   30$                     ; Branch if there are.
                  023F   31     01CC          BRW     999$                    ; Return with status.
                                01CF
     1B    000004BF'EF   91     01CF  30$:    CMPB    TYPEAHD_BUF+2,#ESC      ; Is there an escape sequence in the
                                01D6                                          ;  type ahead buffer?
                    1F   13     01D6          BEQL    READ_WO_PROMPT          ; Branch if so.
  9B 8F    000004BF'EF   91     01D8          CMPB    TYPEAHD_BUF+2,#CSI      ; Is there an 8 bit escape sequence in
                                01E0                                          ;  the type ahead buffer?
                    15   13     01E0          BEQL    READ_WO_PROMPT          ; Branch if so.
     17    000004BF'EF   91     01E2          CMPB    TYPEAHD_BUF+2,#CTRLW    ; Is there a ^W in the type ahead
                                01E9                                          ;  buffer?
                    0C   13     01E9          BEQL    READ_WO_PROMPT          ; Branch if so.
     1A    000004BF'EF   91     01EB          CMPB    TYPEAHD_BUF+2,#CTRLZ    ; Is there a ^Z in the type ahead
                                01F2                                          ;  buffer?
                                01F2          JNEQ    READ_W_PROMPT           ; Branch if not.
                                01F7
                                01F7  READ_WO_PROMPT:
                                01F7          $QIOW_S -                       ; Read the command without prompt.
                                01F7                  CHAN=IN_CHAN,-          ;
                                01F7                  FUNC=#IO$M_EXTEND!IO$_READVBLK,-
                                01F7                  IOSB=IOSB,-             ;
                                01F7                  P1=CMD_BUF,-            ;
                                01F7                  P2=#CMD_BUF_LEN,-       ;
                                01F7                  P5=#TERM_LIST,-         ;
                                01F7                  P6=#TERM_LIST_LEN       ;
```

```
                                  022A         JLBC    R0,999$                  ; Branch on error.
                                  0230
           1A  00002242'EF    91  0230         CMPB    IOSB+4,#CTRLZ            ; Did the user type an control-Z?
                        0C    12  0237         BNEQ    40$                      ; Branch if not.
           50           01    D0  0239         MOVL    #SS$_NORMAL,R0           ; Indicate normal exit.
                                  023C         $EXIT_S                          ; Exit
                                  0245
        9B 8F  00002242'EF    91  0245  40$:   CMPB    IOSB+4,#CSI              ; Did the user type an eight-bit escape
                                  024D                                          ;  sequence?
                        09    13  024D         BEQL    50$                      ; Branch if so.
           1B  00002242'EF    91  024F         CMPB    IOSB+4,#ESC              ; Did the user type an escape sequence?
                        06    12  0256         BNEQ    60$                      ; Branch if not.
                      145E    30  0258  50$:   BSBW    SET_SCROLL               ; Set scroll.
                      FF3B    31  025B         BRW     READ_LOOP                ; Loop till type ahead buffer is empty.
           17  00002242'EF    91  025E  60$:   CMPB    IOSB+4,#CTRLW            ; Did user type control-W?
                                  0265         JLBC    R0,999$                  ; Error if not.
               00000470'EF    94  026B         CLRB    TOP_CMD_LINE             ; Indicate that there is no additional
                                  0271                                          ;  information on the top command line.
  00002248'EF  0000225C'EF    D0  0271         MOVL    CMD_LINE,LINE_NUM        ; Get the number of the last line of
                                  027C                                          ;  the command display area.
                      217E    30  027C         BSBW    ERASE_TWO_LINES          ; Erase any information in the command
                                  027F                                          ;  display area.
                                  027F         JLBC    R0,999$                  ; Branch on error.
      00002348'EF       00    FB  0285         CALLS   #0,REFRESH               ; Otherwise, refresh the screen.
                                  028C         JLBC    R0,999$                  ; Branch on error.
                      FF04    31  0292         BRW     READ_LOOP                ; Loop till type ahead buffer is empty.
                                  0295
                                  0295  READ_W_PROMPT:
  00002248'EF  0000225C'EF    D0  0295         MOVL    CMD_LINE,LINE_NUM        ; Set cursor to line 23,
       06 00002246'EF   01    E1  02A0         BBC     #DVT$V_DCRT,IN_DEV,70$   ;  or if terminal is a DECcrt,
               00002248'EF    D6  02A8         INCL    LINE_NUM                 ;  on line 24.
               0000224C'EF    3F  02AE  70$:   PUSHAW  COL_NUM                  ;
               00002248'EF    3F  02B4         PUSHAW  LINE_NUM                 ;
      00000000'GF       02    FB  02BA         CALLS   #2,G^LIB$SET_CURSOR      ;
                                  02C1         JLBC    R0,999$                  ; Branch on error.
      00000458'EF       01    D0  02C7         MOVL    #1,SCROLL_DELAY          ; Set the number of refreshes before
                                  02CE                                          ;  the command is scrolled.
        09 000004B9'EF        E8  02CE         BLBS    CMD_STATUS,80$           ; Branch if the last command succeeded.
                      1887    30  02D5         BSBW    SCROLL_TWO               ; Scroll the two command lines.
                                  02D8         JLBC    R0,999$                  ; Branch on error.
      00000000'GF       00    FB  02DE  80$:   CALLS   #0,G^LIB$ERASE_LINE      ; Erase the command line.
                                  02E5         JLBC    R0,999$                  ; Branch on error.
       0C 00002246'EF   01    E1  02EB         BBC     #DVT$V_DCRT,IN_DEV,90$   ; Skip the following input device isn't
                                  02F3                                          ;  a DECcrt.
               000019EA'EF    16  02F3         JSB     SET_REGION_COMMAND       ; Establish a scrolling region.
                                  02F9         JLBC    R0,999$                  ; Branch on error.
                                  02FF
                                  02FF  90$:   $QIOW_S -                        ; Read the command with prompt.
                                  02FF                 CHAN=IN_CHAN,-
                                  02FF                 FUNC=#IO$_READPROMPT!IO$M_DSABLMBX,-
                                  02FF                 IOSB=IOSB,-
                                  02FF                 P1=CMD_BUF,-
                                  02FF                 P2=#CMD_BUF_LEN,-
                                  02FF                 P5=#PROMPT+8,-
                                  02FF                 P6=#PROMPT_LEN
                                  0332         JLBC    R0,999$                  ; Branch on error.
                                  0338
```

```
              1A   00002242'EF   91  0338          CMPB    IOSB+4,#CTRLZ             ; Did user type control-Z?
                            0C   12  033F          BNEQ    100$                      ; Branch if not.
                      50    01   D0  0341          MOVL    #SS$_NORMAL,R0            ; Indicate normal exit.
                                     0344          $EXIT_S                           ; Exit
                                     034D
     000020D3'EF   00002240'EF   3C  034D  100$:   MOVZWL  IOSB+2,CMD_LINE_DSCR      ; Save number of characters in command.
                   000020D3'EF   DF  0358          PUSHAL  CMD_LINE_DSCR             ; Convert command line to uppercase.
                   000020D3'EF   DF  035E          PUSHAL  CMD_LINE_DSCR             ;
               00000000'GF  02   FB  0364          CALLS   #2,G^STR$UPCASE           ;
                                     036B          JLBC    R0,999$                   ; Branch on error.
                                     0371
           40 8F   000004BF'EF   91  0371          CMPB    TYPEAHD_BUF+2,#^A/@/      ; Is there a "@" in the type ahead
                                     0379                                            ;  buffer?
                            2C   13  0379          BEQL    110$                      ; If so, bypass DCL_PARSE.
                                     037B
                                     037B  ;
                                     037B  ; Use DCL_PARSE to check for obvious syntax errors.
                                     037B  ;
                                     037B
                   0000040F'EF   DF  037B          PUSHAL  SECOND_CHANCE
                   00000000'EF   DF  0381          PUSHAL  SHOW_CLUSTER$COMMAND_TABLE
                   000020D3'EF   DF  0387          PUSHAL  CMD_LINE_DSCR
               00000000'GF  03   FB  038D          CALLS   #3,G^CLI$DCL_PARSE
                         10 50   E8  0394          BLBS    R0,110$                   ; Branch if no error.
               00000470'EF  04   90  0397          MOVB    #CL$M_ERROR,TOP_CMD_LINE; Indicate that there is error
                                     039E                                            ;  information on both command lines.
               00000471'EF  04   90  039E          MOVB    #CL$M_ERROR,BOT_CMD_LINE;
                            59   11  03A5          BRB     130$                      ;
                                     03A7
                                     03A7  ;
                                     03A7  ; Use TPARSE to do the actual parsing and call the action routines.
                                     03A7  ;
                                     03A7
                                     03A7  ;
                                     03A7  ; Form the descriptor for LIB$TPARSE.
                                     03A7  ;
                                     03A7
     000004CF'EF   000020D3'EF   B0  03A7  110$:   MOVW    CMD_LINE_DSCR,TPARSE_BLK+TPA$L_STRINGCNT
     000004D3'EF   000020DB'EF   DE  03B2          MOVAL   CMD_BUF,TPARSE_BLK+TPA$L_STRINGPTR
                   00000000'EF   DF  03BD          PUSHAL  CMD_KEY                   ; Parse the input line.
                   00000000'EF   DF  03C3          PUSHAL  CMD_STATE                 ;
                   000004C7'EF   DF  03C9          PUSHAL  TPARSE_BLK                ;
               00000000'GF  03   FB  03CF          CALLS   #3,G^LIB$TPARSE           ;
               000004B9'EF  50   D0  03D6          MOVL    R0,CMD_STATUS             ; Save the command status for SIGNAL.
                         0F 50   E9  03DC          BLBC    R0,120$                   ; Branch on error.
               00000470'EF  02   90  03E0          MOVB    #CL$M_CMD,TOP_CMD_LINE    ; Indicate that the last command
                                     03E7                                            ;  entered is now scrolled up one.
                   00000471'EF   94  03E7          CLRB    BOT_CMD_LINE              ;  and the bottom command line is blank.
                            11   11  03ED          BRB     130$                      ; Continue.
               00000470'EF  02   90  03EF  120$:   MOVB    #CL$M_CMD,TOP_CMD_LINE    ; Indicate that the last command
                                     03F6                                            ;  entered is now scrolled up one.
               00000471'EF  04   90  03F6          MOVB    #CL$M_ERROR,BOT_CMD_LINE; And the bottom command line contains
                                     03FD                                            ;  an error message.
                          115D   30  03FD          BSBW    SIGNAL                    ; Output the error.
                                     0400
                                     0400  ;
                                     0400  ; Before looping, make sure ERR_MSGVEC still has the address of TPARSE_BLK+
```

```
                       0400  3081 ; TPA$L_TOKENCNT.  This may have changed if an ADD or REMOVE class was just
                       0400  3082 ; parsed.
                       0400  3083 ;
                       0400  3084
000004D7'EF     DE     0400  3085 130$:   MOVAL   TPARSE_BLK+TPA$L_TOKENCNT,-
000004A9'EF            0406  3086                 ERR_MSGVEC+12             ;
       FD8B     31     040B  3087         BRW     READ_LOOP                 ; Loop till no more command.
                       040E  3088
                05     040E  3089 999$:   RSB
                       040F  3090
                       040F  3091         .DSABL  LSB
                       040F  3092
```

```
                                  040F  ;++
                                  040F  ;
                                  040F  ; FUNCTIONAL DESCRIPTION:
                                  040F  ;
                                  040F  ;       This routine is called when there are parameters or qualifier values
                                  040F  ;       missing from the input string.  When called, DCL_PARSE has already
                                  040F  ;       set up an arguement list to call LIB$GET_INPUT.  The address that
                                  040F  ;       DCL has provided for the return string descriptor is saved and the
                                  040F  ;       call to LIB$GET_INPUT is made.  When the string is returned it is
                                  040F  ;       concatenated with the original command string so that the whole
                                  040F  ;       command can be passed to TPARSE.
                                  040F  ;
                                  040F  ; CALLING SEQUENCE:
                                  040F  ;
                                  040F  ;       CALLS   SECOND_CHANCE
                                  040F  ;
                                  040F  ; CALLED BY:
                                  040F  ;
                                  040F  ;       CLI$DCL_PARSE from READ_W_PROMPT
                                  040F  ;
                                  040F  ; INPUTS:
                                  040F  ;
                                  040F  ;       (AP)    Number of arguments.
                                  040F  ;       (AP)+4  Address of descriptor for string returned.
                                  040F  ;       (AP)+8  Address of prompt descriptor.
                                  040F  ;       (AP)+12 Address to store length of string returned.
                                  040F  ;
                                  040F  ;--
                                  040F
                            003C  040F  .ENTRY  SECOND_CHANCE,^M<R2,R3,R4,R5>
              53      04     D0   0411          MOVL    #4,R3                   ; Get address of DCL's param-routine
              52    6C43     DE   0414          MOVAL   (AP)[R3],R2             ;  string descriptor.
       00000000'GF     6C    FA   0418          CALLG   (AP),G^LIB$GET_INPUT    ; Get missing parameter from user.
                 38    50    E9   041F          BLBC    R0,99$                  ; Branch on error.
   51  000020D3'EF           3C   0422          MOVZWL  CMD_LINE_DSCR,R1        ; Get number of characters in original
                                  0429                                          ;  command line.
   53  000020D7'EF           D0   0429          MOVL    CMD_LINE_DSCR+4,R3      ; Get address of command line.
              53    6341     9E   0430          MOVAB   (R3)[R1],R3             ; Get address of last byte in command.
              83      20     90   0434          MOVB    #^A/ /,(R3)+            ; Change <CR> to <SPACE>.
         51   82      01     C1   0437          ADDL3   #1,(R2)+,R1             ; Get the length of the additional
                                  043B                                          ;  input string (plus the space).
       000020D3'EF     51    A0   043B          ADDW    R1,CMD_LINE_DSCR        ; Save length of combined strings.
        63   00 B2     51    28   0442          MOVC3   R1,@(R2),(R3)           ; Concatenate the two strings.
          000020D3'EF        DF   0447          PUSHAL  CMD_LINE_DSCR           ; Make sure the full line is uppercase.
          000020D3'EF        DF   044D          PUSHAL  CMD_LINE_DSCR           ;
       00000000'GF     02    FB   0453          CALLS   #2,G^STR$UPCASE         ;
                             04   045A  99$:    RET                             ; Return with status.
                                  045B
                                  045B
```

```
                              045B  ;++
                              045B  ;
                              045B  ; FUNCTIONAL DESCRIPTION:
                              045B  ;
                              045B  ;       This routine is entered when a broadcast message AST occurs.  It
                              045B  ;       determines where the end of a line in the broadcast message occurs
                              045B  ;       and then call STORE_MSG to place the broadcast message in a queue
                              045B  ;       to be output to the terminal.  This routine also replaces non-
                              045B  ;       printable characters in the broadcast message with periods (".").
                              045B  ;
                              045B  ; CALLING SEQUENCE:
                              045B  ;
                              045B  ;       JSB     BRDCST
                              045B  ;
                              045B  ; CALLED BY:
                              045B  ;
                              045B  ;       MAIN_LOOP
                              045B  ;
                              045B  ;--
                              045B
             007C 8F     BB   045B  BRDCST: PUSHR   #^M<R2,R3,R4,R5,R6>
    53  00001FCB'EF      DE   045F          MOVAL   MBX_BUF+22,R3           ; Get address of broadcast message.
    52  00001FC9'EF      3C   0466          MOVZWL  MBX_BUF+20,R2           ; Get length of broadcast message.
             54   52     7D   046D  10$:    MOVQ    R2,R4                   ; Save a copy in R4,R5.
                              0470
                              0470  ;
                              0470  ; Scan the message for non-ascii characters.  If a non-ascii character is
                              0470  ; found, substitute a period (".").  Loop until the first carriage return
                              0470  ; following a valid ascii string is found or until the end of the message
                              0470  ; is found.
                              0470  ;
                         CA   0470          BICL    #<MF$M_VALID_MSG!-      ; Clear valid message flag and <CR>
                              0471                  MF$M_CR_FOUND>,-        ;  found after valid message flag.
 00000476'EF  00000600 8F     0471                  MISC_FLAGS              ;
             56   53     D0   047B  20$:    MOVL    R3,R6                   ; Save pointer to broadcast message.
                2265     30   047E          BSBW    CHECK_ASCII             ; Check for non-ascii characters.
             54   50     E8   0481          BLBS    R0,70$                  ; Branch if entire message contains
                              0484                                          ;  valid ascii characters.
             56   53     D1   0484          CMPL    R3,R6                   ; Were any ascii at all characters
                              0487                                          ;  found?
             13   13          0487          BEQL    30$                     ; Branch if not.
                  0A     E0   0489          BBS     #MF$V_CR_FOUND,-        ; Branch if the characters found are
       35 00000476'EF         048B                  MISC_FLAGS,60$          ;  part of the next line of a multi-
                              0491                                          ;  line message.
          00000200 8F    C8   0491          BISL    #MF$M_VALID_MSG,-       ; Set valid message found flag.
          00000476'EF         0497                  MISC_FLAGS
                  52     D7   049C  30$:    DECL    R2                      ; Count down characters remaining in the
                              049E                                          ;  message.
                  38     19   049E          BLSS    70$                     ; Branch if we've reached the end of the
                              04A0                                          ;  message.
             0D   63     91   04A0          CMPB    (R3),#^X0D              ; Was a carriage return found?
                  13     12   04A3          BNEQ    40$                     ; Branch if not.
                  09     E1   04A5          BBC     #MF$V_VALID_MSG,-       ; Branch if a carriage return has been
       0B 00000476'EF         04A7                  MISC_FLAGS,40$          ;  found before any valid ascii
                              04AD                                          ;  characters have been found.
          00000400 8F    C8   04AD          BISL    #MF$M_CR_FOUND,-        ; Indicate a carriage return following
          00000476'EF         04B3                  MISC_FLAGS              ;  a valid message has been found.
```

```
        07   63   91  04B8  40$:   CMPB    (R3),#7             ; Was a BELL found?
             04   12  04BB         BNEQ    50$                 ; Branch if not.
             53   D6  04BD         INCL    R3                  ; Leave the BELL in the buffer
             BA   11  04BF         BRB     20$                 ;  and loop.
        83   2E   90  04C1  50$:   MOVB    #^A/./,(R3)+        ; Substitute a "." for the character.
             B5   11  04C4         BRB     20$                 ; Loop.
                      04C6
                      04C6  ;
                      04C6  ; A carriage return has been found following a string of valid ascii characters
                      04C6  ; indicating the end of a line in a multi-line message.  Calculate the length
                      04C6  ; of the line and store it in the message output queue.
                      04C6  ;
                      04C6
        52   54   7D  04C6  60$:   MOVQ    R4,R2               ; Restore length and address of full
                      04C9                                     ;  message.
   52   56   53   C3  04C9         SUBL3   R3,R6,R2            ; Get length of this line of the
                      04CD                                     ;  message.
             1A   10  04CD         BSB     STORE_MSG           ; Store this line in the message output
                      04CF                                     ;  queue.
   52   54   52   C3  04CF         SUBL3   R2,R4,R2            ; Get the length of the remainder of
                      04D3                                     ;  the message in R2.
        53   56   D0  04D3         MOVL    R6,R3               ; Get the address of the remainder of
                      04D6                                     ;  the message in R3.
             95   11  04D6         BRB     10$                 ; Loop till entire message processed.
                      04D8
                      04D8  ;
                      04D8  ; The end of the message has been found.  Store the last remainder of the
                      04D8  ; message in the message output queue.
                      04D8  ;
                      04D8
        52   54   7D  04D8  70$:   MOVQ    R4,R2               ; Restore length and address of message.
             0C   10  04DB         BSB     STORE_MSG           ; Store the message in the message
                      04DD                                     ;  output queue.
00000458'EF  01   D0  04DD         MOVL    #1,SCROLL_DELAY     ; Set the delay counter.
     007C 8F      BA  04E4         POPR    #^M<R2,R3,R4,R5,R6>
                  05  04E8         RSB
                      04E9
                      04E9
```

```
                                    04E9
                                    04E9  ;++
                                    04E9  ;
                                    04E9  ; FUNCTIONAL DESCRIPTION:
                                    04E9  ;
                                    04E9  ;         Allocate a buffer from the message buffer free queue.  Store a line
                                    04E9  ;         from the broadcast message in the buffer and insert the buffer in the
                                    04E9  ;         message output queue.
                                    04E9  ;
                                    04E9  ; CALLING SEQUENCE:
                                    04E9  ;
                                    04E9  ;         JSB     STORE_MSG
                                    04E9  ;
                                    04E9  ; CALLED BY:
                                    04E9  ;
                                    04E9  ;         BRDCST
                                    04E9  ;
                                    04E9  ; IMPLICIT INPUTS:
                                    04E9  ;
                                    04E9  ;         MSG_FRE_QUE contains a queue of buffers allocated and available for use.
                                    04E9  ;         MSG_OUT_QUE contains a queue of broadcast message lines to be output.
                                    04E9  ;
                                    04E9  ; INPUTS:
                                    04E9  ;
                                    04E9  ;         R2 contains the length of the message.
                                    04E9  ;         R3 contains the address of the message.
                                    04E9  ;
                                    04E9  ; IMPLICIT OUTPUTS:
                                    04E9  ;
                                    04E9  ;         Message line inserted at end of MSG_OUT_QUE
                                    04E9  ;
                                    04E9  ;--
                                    04E9
                                    04E9  STORE_MSG:
 00000000'EF   00000008'FF    0F    04E9          REMQUE  @MSG_FRE_QUE,BUFADR     ; Get a from the message buffer free
                                    04F4                                          ;  queue.
                       16     1C    04F4          BVC     10$                     ; Branch if buffer available.
               00000000'EF    9F    04F6          PUSHAB  BUFADR                  ; Allocate a new buffer.
               00000004'EF    9F    04FC          PUSHAB  BUFSIZE                 ;
          00000000'GF   02    FB    0502          CALLS   #2,G^LIB$GET_VM         ;
                    21  50    E9    0509          BLBC    R0,99$                  ; Branch on error.
                        3C    BB    050C  10$:    PUSHR   #^M<R2,R3,R4,R5>
    54    00000000'EF   08    C1    050E          ADDL3   #8,BUFADR,R4            ; Get the address to store the message.
                    84  52    B0    0516          MOVW    R2,(R4)+                ; Store message length in the buffer.
              64    63  52    28    0519          MOVC3   R2,(R3),(R4)            ; Store the message in the buffer.
                        3C    BA    051D          POPR    #^M<R2,R3,R4,R5>
 00000014'FF   00000000'FF    0E    051F          INSQUE  @BUFADR,@MSG_OUT_QUE+4  ; Insert buffer into broadcast message
                                    052A                                          ;  output queue.
                    50  01    D0    052A          MOVL    #SS$_NORMAL,R0
                              05    052D  99$:    RSB                             ; Return with status.
                                    052E
                                    052E
```

```
                          052E
                          052E  ;++
                          052E  ;
                          052E  ; FUNCTIONAL DESCRIPTION:
                          052E  ;
                          052E  ;       This routine performs the allocation and deallocation of variable
                          052E  ;       length buffers used by the SNAPSHOT routine.  No more than two
                          052E  ;       buffers are ever allocated.  Before taking a new snapshot, the
                          052E  ;       buffer used for the last snapshot becomes the "old" buffer.  This
                          052E  ;       is done by simply moving the new buffer descriptor to the old buffer
                          052E  ;       descriptors (see SAVE_OLD routine).  The previous "old" buffer is then
                          052E  ;       deallocated.  What are now the "old" buffer descriptors are used to
                          052E  ;       determine how much space to allocate for the new buffer based on the
                          052E  ;       following:
                          052E  ;
                          052E  ;                   OLD                                    New
                          052E  ;     ----------------------------               ------------------------------
                          052E  ;      n System Blocks                           n+4 System Blocks
                          052E  ;      x Cluster System Blocks                   n+4 Cluster System Blocks
                          052E  ;      y Path Blocks                             (n+4)*4 Path Blocks
                          052E  ;      z Connection Descriptors                  (n+4)*4+z Connection Descriptors
                          052E  ;
                          052E  ;       Where n,x,y, and z are the number of blocks found by the last snapshot
                          052E  ;       routine.  (not the number last allocated)
                          052E  ;
                          052E  ;       Once the size of the buffer is determined, the buffer is allocated and
                          052E  ;       locked in memory along with the fixed length buffers and other data
                          052E  ;       needed by the snapshot routine.  A snapshot is then taken.  After the
                          052E  ;       snapshot, the number of entries of each of the four types shown above
                          052E  ;       is determined and saved for the next buffer allocation.
                          052E  ;
                          052E  ; CALLING SEQUENCE:
                          052E  ;
                          052E  ;       JSB     DO_SNAPSHOT
                          052E  ;
                          052E  ; CALLED BY:
                          052E  ;
                          052E  ;       DISPLAY
                          052E  ;
                          052E  ; OUTPUTS:
                          052E  ;
                          052E  ;--
                          052E
                          052E  DO_SNAPSHOT:
                3C    BB  052E          PUSHR   #^M<R2,R3,R4,R5>
   000003E4'EF  02    91  0530  10$:    CMPB    #2,BUF_COUNT            ; Have we allocated two variable length
                          0537                                          ;  buffers?
                08    13  0537          BEQL    20$                     ; Branch if so.
       000003E4'EF    96  0539          INCB    BUF_COUNT               ; Increase count of buffers allocated.
                19    11  053F          BRB     30$                     ; Jump to common code.
       00000419'EF    DF  0541  20$:    PUSHAL  OLD_VAR_BUF             ; Free up the old buffer before we
       00000421'EF    DF  0547          PUSHAL  OLD_VAR_BUF_LEN         ;  allocate a new one.
   00000000'GF  02    FB  054D          CALLS   #2,G^LIB$FREE_VM        ;
                          0554          JLBC    R0,99$                  ; Branch on error.
                          055A
              02D8    30  055A  30$:    BSBW    SAVE_OLD                ; Replace old descriptors with new
                          055D                                          ;  descriptors.
```

```
                51   000003FB'EF   B0 055D         MOVW    OLD_SB_DSCR+2,R1           ; Get the number of system blocks found
                                      0564                                            ;  in the last snapshot.
                         51   04   A0 0564         ADDW    #4,R1                      ; Increase by four for expansion.
              00000029'EF     51   B0 0567         MOVW    R1,SB_DSCR+2               ; Set number of system blocks that will
                                      056E                                            ;  fit in the buffer.
   00000027'EF     51   0060 8F    A5 056E         MULW3   #SB$C_LENGTH,R1,SB_DSCR    ; Set the length of the system block
                                      0578                                            ;  buffer.
              00000031'EF     51   B0 0578         MOVW    R1,CSB_DSCR+2              ; Set the number of cluster system
                                      057F                                            ;  blocks that will fit in the buffer.
   0000002F'EF     51   00AC 8F    A5 057F         MULW3   #CSB$C_LENGTH,R1,-         ; Set the length of the cluster system
                                      0589                 CSB_DSCR                   ;  block buffer.
                         51   04   A4 0589         MULW    #4,R1                      ; Get the number of path blocks that
                                      058C                                            ;  will fit in the buffer.
              00000039'EF     51   B0 058C         MOVW    R1,PB_DSCR+2               ; Save in descriptor.
   00000037'EF     51   0054 8F    A5 0593         MULW3   #PB$C_LENGTH,R1,PB_DSCR    ; Set the length of the path block
                                      059D                                            ;  buffer.
                51   00000413'EF   A0 059D         ADDW    OLD_CDT_DSCR+2,R1          ; Allow number of connections to expand
                                      05A4                                            ;  by one per path.
              00000041'EF     51   B0 05A4         MOVW    R1,CDT_DSCR+2              ; Save in descriptor.
                   51   008C 8F    A4 05AB         MULW    #SHC$C_CDT_LENGTH,R1       ; Get the length of the connections
                                      05B0                                            ;  buffer.
              0000003F'EF     51   B0 05B0         MOVW    R1,CDT_DSCR                ; Save in descriptor.
     00000419'EF    000003ED'EF    D0 05B7         MOVL    VAR_BUF,OLD_VAR_BUF        ; Save address of previous buffer.
                    000003F5'EF    D0 05C2         MOVL    VAR_BUF_LEN,-              ; Save length of previous buffer.
                    00000421'EF       05C8                 OLD_VAR_BUF_LEN            ;
                                      05CD ;
                                      05CD ; Total the number of bytes needed for the variable length buffers.
                                      05CD ;
                                      05CD
              000003F5'EF     51   3C 05CD         MOVZWL  R1,VAR_BUF_LEN             ; CDT buffer.
     000003F5'EF    00000027'EF    A0 05D4         ADDW    SB_DSCR,VAR_BUF_LEN        ; SB buffer.
                             0B    1E 05DF         BCC     40$
     000003F5'EF    00010000'EF    C0 05E1         ADDL    ^X10000,VAR_BUF_LEN
     000003F5'EF    0000002F'EF    A0 05EC 40$:    ADDW    CSB_DSCR,VAR_BUF_LEN       ; CSB buffer.
                             0B    1E 05F7         BCC     50$
     000003F5'EF    00010000'EF    C0 05F9         ADDL    ^X10000,VAR_BUF_LEN
     000003F5'EF    00000037'EF    A0 0604 50$:    ADDW    PB_DSCR,VAR_BUF_LEN        ; PB buffer.
                             0B    1E 060F         BCC     60$
     000003F5'EF    00010000'EF    C0 0611         ADDL    ^X10000,VAR_BUF_LEN
                                      061C
                                      061C ;
                                      061C ; Save the number of entries we can store in each of the buffers.
                                      061C ; The numbers will be used to determine how many entries were actually
                                      061C ; stored in each of the buffers.
                                      061C ;
                                      061C
     000003E5'EF    00000029'EF    B0 061C 60$:    MOVW    SB_DSCR+2,SB_COUNT         ; Number of entrys that can be stored
                                      0627                                            ;  in the system block buffer.
     000003E7'EF    00000031'EF    B0 0627         MOVW    CSB_DSCR+2,CSB_COUNT       ; Number of entrys that can be stored
                                      0632                                            ;  in the cluster system block buffer.
     000003E9'EF    00000039'EF    B0 0632         MOVW    PB_DSCR+2,PB_COUNT         ; Number of entrys that can be stored
                                      063D                                            ;  in the path block buffer.
     000003EB'EF    00000041'EF    B0 063D         MOVW    CDT_DSCR+2,CDT_COUNT       ; Number of entrys that can be stored
                                      0648                                            ;  in the CDT buffer.
                                      0648
                    000003ED'EF    DF 0648         PUSHAL  VAR_BUF                    ; Allocate some buffer space.
                    000003F5'EF    DF 064E         PUSHAL  VAR_BUF_LEN                ;
```

```
          00000000'GF    02   FB 0654          CALLS   #2,G^LIB$GET_VM            ;
                                  065B          JLBC    R0,99$                     ; Branch on error.
      00000047'EF  000003ED'EF   D0 0661          MOVL    VAR_BUF,VAR_BUF_ADRS       ; Save copy of buffer address for
                                  066C                                             ;  snapshot routine.
      000003F5'EF  000003ED'EF   C1 066C          ADDL3   VAR_BUF,VAR_BUF_LEN,-      ; Calculate address of end of buffer.
                   000003F1'EF      0677                  VAR_BUF_END                ;
                                  067C
        00  000003ED'FF    00   2C 067C          MOVC5   #0,@VAR_BUF,#0,-           ; Zero out the variable length snapshot
       000003ED'FF    03F5'8F      0684                  #VAR_BUF_LEN,@VAR_BUF      ;  buffers.
                                  068C
        00  0000004B'EF    00   2C 068C          MOVC5   #0,FIX_BUF,#0,-            ; Zero out the fixed length snapshot
       0000004B'EF    0458'8F      0694                  #FIX_BUF_LEN,FIX_BUF       ;  buffers.
                                  069C
                                  069C ;
                                  069C ; Complete the variable buffer descriptors.
                                  069C ;
                                  069C
      0000002B'EF  000003ED'EF   D0 069C          MOVL    VAR_BUF,SB_DSCR+4          ; Save address of system block buffer.
              51  00000027'EF   3C 06A7          MOVZWL  SB_DSCR,R1                 ; Get length of system block buffer.
00000033'EF  0000002B'EF    51   C1 06AE          ADDL3   R1,SB_DSCR+4,CSB_DSCR+4    ; Save address of cluster system block
                                  06BA                                             ;  buffer.
              51  0000002F'EF   3C 06BA          MOVZWL  CSB_DSCR,R1                ; Get length of cluster system block
                                  06C1                                             ;  buffer.
0000003B'EF  00000033'EF    51   C1 06C1          ADDL3   R1,CSB_DSCR+4,PB_DSCR+4    ; Save address of path block buffer.
              51  00000037'EF   3C 06CD          MOVZWL  PB_DSCR,R1                 ; Get length of path block buffer.
00000043'EF  0000003B'EF    51   C1 06D4          ADDL3   R1,PB_DSCR+4,CDT_DSCR+4    ; Save address of CDT buffer.
                                  06E0
                                  06E0 ;
                                  06E0 ; Set up buffer pointers.
                                  06E0 ;
      00000017'EF  0000002B'EF   D0 06E0          MOVL    SB_DSCR+4,SB_BUF_PNTR      ; Initialize pointer into system block
                                  06EB                                             ;  buffer.
      0000001B'EF  00000033'EF   D0 06EB          MOVL    CSB_DSCR+4,CSB_BUF_PNTR    ; Initialize pointer into cluster
                                  06F6                                             ;  system block buffer.
      0000001F'EF  0000003B'EF   D0 06F6          MOVL    PB_DSCR+4,PB_BUF_PNTR      ; Initialize pointer into path block
                                  0701                                             ;  buffer.
      00000023'EF  00000043'EF   D0 0701          MOVL    CDT_DSCR+4,CDT_BUF_PNTR    ; Initialize pointer into connection
                                  070C                                             ;  descriptor buffer.
                                  070C
                                  070C          $LKWSET_S -                        ; Lock variable length snapshot buffers
                                  070C                  INADR=VAR_BUF              ;  in memory.
                                  071D          JLBC    R0,99$                     ; Branch on error.
                                  0723
                                  0723          $LKWSET_S -                        ; Lock fixed snapshot buffers and
                                  0723                  INADR=SNAP_ADRS            ;  snapshot routine in memory.
                                  0734          JLBC    R0,99$                     ; Branch on error.
                                  073A
                                  073A          $CMKRNL_S -                        ; Capture SCS database.
                                  073A                  ROUTIN=SNAPSHOT            ;
          000004A1'EF    50   D0 0749          MOVL    R0,ERR_MSGVEC+4            ; Save snapshot status.
                                  0750
                                  0750          $ULWSET_S -                        ; Unlock snapshot buffers and snapshot
                                  0750                  INADR=SNAP_ADRS            ;  routine pages.
                                  0761          JLBC    R0,99$                     ; Branch on error.
                                  0767
                                  0767          $ULWSET_S -                        ; Unlock variable snapshot buffer pages.
                                  0767                  INADR=VAR_BUF              ;
```

```
                                   0778             JLBC    R0,99$                  ; Branch on error.
                                   077E
        50  000004A1'EF      D0   077E             MOVL    ERR_MSGVEC+4,R0          ; Restore snapshot status.
                                   0785             JLBC    R0,99$                  ; Branch on error.
                                   078B       ;
                                   078B       ; Calculate the number of entries actually stored in each of the variable
                                   078B       ; length buffers by subtacting the number of empty entries remaining from
                                   078B       ; the number of entries we started with.
                                   078B       ;
                                   078B
 000003E5'EF  00000029'EF    A3   078B             SUBW3   SB_DSCR+2,SB_COUNT,-      ; Number of system block entries in
              00000029'EF         0796                     SB_DSCR+2                 ;  buffer.
 000003E7'EF  00000031'EF    A3   079B             SUBW3   CSB_DSCR+2,CSB_COUNT,-    ; Number of cluster system block
              00000031'EF         07A6                     CSB_DSCR+2                ;  entries in buffer.
 000003E9'EF  00000039'EF    A3   07AB             SUBW3   PB_DSCR+2,PB_COUNT,-      ; Number of path block entries in
              00000039'EF         07B6                     PB_DSCR+2                 ;  buffer.
 000003EB'EF  00000041'EF    A3   07BB             SUBW3   CDT_DSCR+2,CDT_COUNT,-    ; Number of CDT entries in buffer.
              00000041'EF         07C6                     CDT_DSCR+2                ;
                                   07CB
                          00 E4   07CB             BBSC    #MF$V_INIT_FLAG,-         ; Old = new if init pass in progress.
            08 00000476'EF        07CD                     MISC_FLAGS,70$            ;
         57 00000476'EF   03 E1   07D3             BBC     #MF$V_RIP,MISC_FLAGS,99$; Or if refresh just performed.
                          58 10   07DB       70$:  BSB     SAVE_OLD                  ;
     0000004B'EF   0458'8F 2C     07DD             MOVC5   #FIX_BUF_LEN,FIX_BUF,-    ; Duplicate the fixed length buffers so
             0458'8F      00      07E6                     #0,#FIX_BUF_LEN,-         ;  we don't display the data in inverse
              000004A3'EF         07EA                     OLD_FIX_BUF               ;  video.
                                   07EF
                                   07EF       ;
                                   07EF       ; Check if we got the full data base on this snapshot.  If so go display the
                                   07EF       ; report.  Otherwise, loop to expand buffers and try again.
                                   07EF       ;
                                   07EF
 000003E5'EF  00000029'EF    B1   07EF             CMPW    SB_DSCR+2,SB_COUNT        ; Did we run out of system block buffer?
                                   07FA             JEQLU   10$                       ; Branch if so.
 000003E7'EF  00000031'EF    B1   07FF             CMPW    CSB_DSCR+2,CSB_COUNT      ; Did we run out of clustre system
                                   080A                                               ;  block buffer?
                                   080A             JEQLU   10$                       ; Branch if so.
 000003E9'EF  00000039'EF    B1   080F             CMPW    PB_DSCR+2,PB_COUNT        ; Did we run out of path block buffer?
                                   081A             JEQLU   10$                       ; Branch if so.
 000003EB'EF  00000041'EF    B1   081F             CMPW    CDT_DSCR+2,CDT_COUNT      ; Did we run out of connection
                                   082A                                               ;  descriptor buffer?
                                   082A             JEQLU   10$                       ; Branch if so.
                                   082F
                     50   01 D0   082F             MOVL    #SS$_NORMAL,R0            ; Indicate success.
                                   0832
                          3C BA   0832       99$:  POPR    #^M<R2,R3,R4,R5>
                             05   0834             RSB                               ; Return with status.
                                   0835
                                   0835
```

```
                                  0835  ;++
                                  0835  ;
                                  0835  ; FUNCTIONAL DESCRIPTION:
                                  0835  ;
                                  0835  ;       This routine performs two functions:
                                  0835  ;               1) When called prior to taking a snapshot it is used to move the
                                  0835  ;                  newest set of buffer descriptors to the storage area for the
                                  0835  ;                  old buffer descriptors before building a new set.
                                  0835  ;               2) When called after taking a snapshot it is used to duplicate
                                  0835  ;                  the new buffer descriptors so that both sets point to the
                                  0835  ;                  same buffers.  This will mean that the CHECK_DIF routine.
                                  0835  ;                  will extract the same data when comparing new and old data.
                                  0835  ;
                                  0835  ; CALLING SEQUENCE:
                                  0835  ;
                                  0835  ;       JSB     SAVE_OLD
                                  0835  ;
                                  0835  ; CALLED BY:
                                  0835  ;
                                  0835  ;       DO_SNAPSHOT
                                  0835  ;
                                  0835  ;--
                                  0835
                                  0835  SAVE_OLD:
000003F9'EF 00000027'EF  7D  0835          MOVQ    SB_DSCR,OLD_SB_DSCR     ; Make descriptor for variable length
00000401'EF 0000002F'EF  7D  0840          MOVQ    CSB_DSCR,OLD_CSB_DSCR   ;   buffers used in the last pass point
00000409'EF 00000037'EF  7D  084B          MOVQ    PB_DSCR,OLD_PB_DSCR     ;   to the ones for this pass so we don't
00000411'EF 0000003F'EF  7D  0856          MOVQ    CDT_DSCR,OLD_CDT_DSCR   ;   display the data in inverse video.
                         05  0861          RSB
                             0862
```

```
                     0862        .SBTTL  Initialization.
                 00000862        .PSECT  MAIN,RD,EXE,NOWRT
                     0862  ;++
                     0862  ;
                     0862  ; FUNCTIONAL DESCRIPTION:
                     0862  ;
                     0862  ;       This routine performs the following steps for program initialization.
                     0862  ;
                     0862  ;       - Establishes an exit handler.
                     0862  ;       - Assigns channels to I/O devices.
                     0862  ;       - Determines the device type of the I/O devices and saves a summary of
                     0862  ;         the type.
                     0862  ;       - Translates logical name SYS$NODE.
                     0862  ;       - Saves the system ID of the local system.
                     0862  ;       - Saves a copy of the display bitmap for use by the INITIALIZE command.
                     0862  ;       - Parses the DCL command line for /REPORT=, /CONTINUOUS, /INTERVAL=,
                     0862  ;         /OUTPUT= and sets up accordingly.
                     0862  ;       - Executes the startup initialization file.
                     0862  ;       - Establishes a control-C handler.
                     0862  ;       - Sets the first AST for unsolicited data.
                     0862  ;       - If input is from a terminal:
                     0862  ;               - A mailbox is associated with the terminal.
                     0862  ;               - New terminal characteristics are set up after saving the old
                     0862  ;                 ones for restoration upon exit.
                     0862  ;               - Screen is cleared if terminal is a scope.
                     0862  ;
                     0862  ; CALLING SEQUENCE:
                     0862  ;
                     0862  ;       JSB     INIT
                     0862  ;
                     0862  ; CALLED BY:
                     0862  ;
                     0862  ;       START
                     0862  ;
                     0862  ; OUTPUTS:
                     0862  ;
                     0862  ;       IN_DEV, OUT_DEV Set to reflect the I/O device type.
                     0862  ;       SAVE_CHAR_BUF   Contains the original terminal characteristics.
                     0862  ;       LOCSYSID        Receives the system ID of the local system.
                     0862  ;       CL_SAVE_BITMAP  Receives a copy of the cluster bitmap.
                     0862  ;       LOC_SAVE_BITMAP Reveives a copy of the local bitmap.
                     0862  ;
                     0862  ; RETURN STATUS:
                     0862  ;
                     0862  ;       SS$_NORMAL or failed status returned in R0.
                     0862  ;
                     0862  ;--
                     0862
00FC 8F   BB         0862  INIT:   PUSHR   #^M<R2,R3,R4,R5,R6,R7>
                     0866
                     0866  ;
                     0866  ; Setup an exit handler.
                     0866  ;
                     0866
                     0866          $DCLEXH_S -                     ; Declare exit handler.
                     0866                  DESBLK=EXIT_BLK         ;
                     0873          JLBC    R0,99$                  ; Branch on error.
```

```
                                  0879
                                  0879       $GETJPI_S -                      ; Get mask of the user's current
                                  0879            ITMLST=JPI_LIST             ;  privileges.
                                  0890       JLBC    R0,99$                   ; Branch on error.
                                  0896
           00002230'EF   00   E0  0896       BBS     #PRV$V_CMKRNL,CUR_PRIV,-; Branch if user has change mode KERNEL
                         0A       089D               4$                       ;  privilege.
        50   00002804 8F      D0  089E       MOVL    #SS$_NOCMKRNL,R0         ; Indicate CMKRNL required.
                       066E   31  08A5       BRW     99$                      ; Exit with error status.
                                  08A8
                                  08A8 4$:   $GETDVI_S -                      ; Get the input device characteristics.
                                  08A8            DEVNAM=IN_DEV_NAME,-        ;
                                  08A8            ITMLST=GET_LIST             ;
                                  08C6       JLBC    R0,99$                   ; Branch on error.
                                  08CC
       00002208'EF    42 8F   91  08CC       CMPB    #DC$_TERM,DEVICE_CLASS   ; Is input device a terminal?
                         08   13  08D4       BEQL    5$                       ; Branch if so.
       7B 00002246'EF    05   E2  08D6       BBSS    #DVT$V_DISK,IN_DEV,30$   ; Input is from a disk file.
                                  08DE
              0000223C'EF     DF  08DE 5$:   PUSHAL  MBXCHAN                  ; Assign a channel to the terminal and
              00002234'EF     DF  08E4       PUSHAL  IN_CHAN                  ;  associate a mailbox with it.
              000004B5'EF     DF  08EA       PUSHAL  BUF_QUO                  ;
              000004B1'EF     DF  08F0       PUSHAL  MAX_MSG                  ;
              000000DE'EF     DF  08F6       PUSHAL  IN_DEV_NAME              ;
           00000000'GF   05   FB  08FC       CALLS   #5,G^LIB$ASN_WTH_MBX     ;
                                  0903       JLBC    R0,99$                   ; Branch on error.
                                  0909
                                  0909 ;
                                  0909 ; Check the input terminal characteristics and summarize in IN_DEV.
                                  0909 ;
 0000221C'EF    20000000 8F   D3  0909       BITL    #TT2$M_DECCRT,DEVDEPEND2; Is terminal a DECcrt?
                         09   13  0914       BEQL    7$                       ; Branch if not.
           00002246'EF   02   88  0916       BISB    #DVT$M_DCRT,IN_DEV       ; Set VT100 bit.
                         3A   11  091D       BRB     30$                      ; Jump to common code.
 0000220C'EF    00000040 8F   D3  091F 7$:   BITL    #DT$_VT5X,DEVICE_TYPE    ; Is terminal VT52 compatable?
                         09   13  092A       BEQL    10$                      ; Branch if not.
           00002246'EF   01   88  092C       BISB    #DVT$M_VT52,IN_DEV       ; Set VT52 bit.
                         24   11  0933       BRB     30$                      ; Jump to common code.
 00002218'EF    00001000 8F   D3  0935 10$:  BITL    #TT$M_SCOPE,DEVDEPEND    ; Is the terminal some type of scope?
                         10   13  0940       BEQL    20$                      ; Branch if not.
           00002246'EF   01   88  0942       BISB    #DVT$M_VT52,IN_DEV       ; Treat terminal as if its a VT52.
           00002246'EF   08   88  0949       BISB    #DVT$M_UNKNOWN,IN_DEV    ; Indicate actual type is unknown.
                         07   11  0950       BRB     30$                      ; Jump to common code.
           00002246'EF   04   88  0952 20$:  BISB    #DVT$M_HARD,IN_DEV       ; Assume terminal is some type of
                                  0959                                        ;  hardcopy device.
                                  0959
                                  0959 ;
                                  0959 ; Get the output terminal characteristics and summarize in OUT_DEV.  Save
                                  0959 ; a copy of the characteristics for restoration upon exit.  Set new
                                  0959 ; characteristics NOBRDCST and BRDCSTMBX.
                                  0959 ;
                                  0959
              0000004B'EF     DF  0959 30$:  PUSHAL  OUTPUT_QUAL              ; Was /OUTPUT specified?
           00000000'GF   01   FB  095F       CALLS   #1,G^CLI$PRESENT         ;
                      12 50   E8  0966       BLBS    R0,31$                   ; Branch if so.
                                  0969
              00000101'EF     28  0969       MOVC3   NULL_DEV_NAME,-          ; Otherwise, NL: is the output device.
```

```
             00000109'EF           096F                  NULL_DEV_NAME+8,-          ;
             00000224'EF           0974                  OUT_FILE_SPEC              ;
                      2F    11     0979          BRB     32$                        ; Branch to common code.
                                   097B
             00001DAB'EF    DF     097B  31$:    PUSHAL  RET_BUF_DSCR               ; Get the output qualifier.
             00001DA3'EF    DF     0981          PUSHAL  GET_VAL_DSCR               ;
             0000004B'EF    DF     0987          PUSHAL  OUTPUT_QUAL                ;
        00000000'GF   03    FB     098D          CALLS   #3,G^CLI$GET_VALUE         ;
                                   0994          JLBC    R0,99$                     ; Branch on error.
                                   099A
00001DB3'EF  00001DAB'EF    28     099A          MOVC3   RET_BUF_DSCR,RET_BUF,-     ; Store the output file string for
             00000224'EF           09A5                  OUT_FILE_SPEC              ;  $PARSE.
                                   09AA
                                   09AA  32$:    $PARSE  FAB = OUTFAB               ; Parse the filespec.
                                   09B7          JLBC    R0,99$                     ; Branch on error.
                                   09BD          JBS     #DEV$V_NET,-               ; Branch if network device.
                                   09BD                  OUTFAB+FAB$L_DEV,57$       ;
                                   09CC          JBC     #DEV$V_TRM,-               ; Branch if output device isn't a
                                   09CC                  OUTFAB+FAB$L_DEV,55$       ;  terminal.
             000001D8'EF    9B     09DB          MOVZBW  OUTNAM_BLOCK+NAM$T_DVI,-; Build a descriptor for GETDVI.
             00001DAB'EF           09E1                  RET_BUF_DSCR               ;
             00001DAB'EF    28     09E6          MOVC3   RET_BUF_DSCR,-             ;
             000001D9'EF           09EC                  OUTNAM_BLOCK+NAM$T_DVI+1,- ;
             00001DB3'EF           09F1                  RET_BUF                    ;
                                   09F6          $GETDVI_S -                        ; Get the output device characteristics.
                                   09F6                  DEVNAM=RET_BUF_DSCR,-      ;
                                   09F6                  ITMLST=GET_LIST            ;
                   06 50    E8     0A14  33$:    BLBS    R0,34$                     ; Branch if successful.
                   50 01    D0     0A17          MOVL    #SS$_NORMAL,R0             ; Otherwise treat it like a file.
                    0123    31     0A1A          BRW     55$                        ;
                                   0A1D
   00002208'EF      42 8F   91     0A1D  34$:    CMPB    #DC$_TERM,DEVICE_CLASS     ; Is output device a terminal?
                                   0A25          JNEQ    55$                        ; Branch if not.
                                   0A2A
                                   0A2A          $ASSIGN_S -                        ; Assign a channel to the output device.
                                   0A2A                  DEVNAM=OUT_DEV_NAME,-      ;
                                   0A2A                  CHAN=OUT_CHAN              ;
                                   0A3F          JLBC    R0,99$                     ; Branch on error.
                                   0A45
0000221C'EF    20000000 8F  D3     0A45          BITL    #TT2$M_DECCRT,DEVDEPEND2; Is output device a DECcrt?
                      1D    13     0A50          BEQL    35$                        ; Branch if not.
         00002247'EF  02    88     0A52          BISB    #DVT$M_DCRT,OUT_DEV        ; Set VT100 bit.
0000221C'EF    08000000 8F  D3     0A59          BITL    #TT2$M_AVO,DEVDEPEND2      ; Does the DECcrt have the advanced
                                   0A64                                             ;  video option?
                      43    13     0A64          BEQL    50$                        ; Branch if not.
         00002247'EF  10    88     0A66          BISB    #DVT$M_AVO,OUT_DEV         ; Set advanced video bit.
                      3A    11     0A6D          BRB     50$                        ; Jump to common code.
0000220C'EF    00000040 8F  D3     0A6F  35$:    BITL    #DT$_VT5X,DEVICE_TYPE      ; Is output device VT52 compatable?
                      09    13     0A7A          BEQL    40$                        ; Branch if not.
         00002247'EF  01    88     0A7C          BISB    #DVT$M_VT52,OUT_DEV        ; Set VT52 bit.
                      24    11     0A83          BRB     50$                        ; Jump to common code.
00002218'EF    00001000 8F  D3     0A85  40$:    BITL    #TT$M_SCOPE,DEVDEPEND      ; Is the output device some type of
                                   0A90                                             ;  scope?
                      10    13     0A90          BEQL    45$                        ; Branch if not.
         00002247'EF  01    88     0A92          BISB    #DVT$M_VT52,OUT_DEV        ; Treat output device as if its a VT52.
         00002247'EF  08    88     0A99          BISB    #DVT$M_UNKNOWN,OUT_DEV     ; Indicate actual type is unknown.
                      07    11     0AA0          BRB     50$                        ; Jump to common code.
```

```
           00002247'EF     04   88  0AA2  45$:   BISB    #DVT$M_HARD,OUT_DEV       ; Assume output device is some type of
                                    0AA9                                           ;  hardcopy device.
                                    0AA9  ;
                                    0AA9  ; Build the characteristics buffer for setmode so that the terminal
                                    0AA9  ; characteristics can be restored.
                                    0AA9  ;
                                    0AA9
                  00002208'EF  90   0AA9  50$:   MOVB    DEVICE_CLASS,-            ; Store the device class.
                  00002214'EF       0AAF                 SAVE_CHAR_BUF             ;
                  0000220C'EF  90   0AB4         MOVB    DEVICE_TYPE,-             ; Store the device type.
                  00002215'EF       0ABA                 SAVE_CHAR_BUF+1           ;
                  00002210'EF  B0   0ABF         MOVW    BUFFER_SIZE,-             ; Store the terminal width.
                  00002216'EF       0AC5                 SAVE_CHAR_BUF+2           ;
                                    0ACA
     00002260'EF  0000221B'EF  9A   0ACA         MOVZBL  DEVDEPEND+3,LAST_LINE     ; Save the number of the last line
                                    0AD5                                           ;  (bottom command line) on the
                                    0AD5                                           ;  terminal.
                                    0AD5  ;
                                    0AD5  ; Setup new terminal characteristics for output device.
                                    0AD5  ;
                                    0AD5
                  00002214'EF  7D   0AD5         MOVQ    SAVE_CHAR_BUF,-           ; Get a copy of the terminal
                  00002220'EF       0ADB                 NEW_CHAR_BUF              ;  characteristics for new setup.
                  0000221C'EF  D0   0AE0         MOVL    DEVDEPEND2,-              ;
                  00002228'EF       0AE6                 NEW_CHAR_BUF+8            ;
                                    0AEB
                  00020008 8F  C8   0AEB         BISL    #TT$M_NOBRDCST!TT$M_ESCAPE,-
                  00002224'EF       0AF1                 NEW_CHAR_BUF+4            ; Set no broadcast bit and escape bit
                                    0AF6                                           ;  in terminal characteristics.
                  00000200 8F  CA   0AF6         BICL    #TT$M_WRAP,-              ; Clear wrap bit in terminal
                  00002224'EF       0AFC                 NEW_CHAR_BUF+4            ;  characteristics.
                           10  C8   0B01         BISL    #TT2$M_BRDCSTMBX,-        ; Set broadcast mailbox bit in extended
                  00002228'EF       0B03                 NEW_CHAR_BUF+8            ;  terminal characteristics.
                                    0B08
                                    0B08         $QIOW_S -                         ; Set new characteristics.
                                    0B08                 CHAN=OUT_CHAN,-           ;
                                    0B08                 FUNC=#IO$_SETMODE,-       ;
                                    0B08                 P1=NEW_CHAR_BUF,-         ;
                                    0B08                 P2=#CHAR_BUF_LEN          ;
                                    0B2D         JLBC    R0,99$                    ; Branch on error.
                  00000080 8F  C8   0B33         BISL    #MF$M_NEW_CHAR,-          ; Indicate that the terminal
                  00000476'EF       0B39                 MISC_FLAGS                ;  characteristics have been changed.
                                    0B3E
                           34  11   0B3E         BRB     60$                       ; Jump to common code.
                                    0B40
                  00000274'EF  DF   0B40  55$:   PUSHAL  OUT_EXP_DSCR              ; Redirect output to file.
                  00002238'EF  DF   0B46         PUSHAL  SET_CHAN                  ;
           00000000'GF     02  FB   0B4C         CALLS   #2,G^LIB$SET_OUTPUT       ;
                                    0B53         JLBC    R0,99$                    ; Branch on error.
                                    0B59
                           24  88   0B59  57$:   BISB    #DVT$M_DISK!DVT$M_HARD,-; Output is to a file.
                  00002247'EF       0B5B                 OUT_DEV                   ;
        00002222'EF   0084 8F  B0   0B60         MOVW    #132,SCREEN_WIDTH         ; Assume 132 columns.
     00002260'EF  00000044 8F  D0   0B69         MOVL    #68,LAST_LINE             ; Assume 68 lines.
                                    0B74
0000225C'EF  00002260'EF   01  C3   0B74  60$:   SUBL3   #1,LAST_LINE,CMD_LINE     ; Save the number of the next-to-last
                                    0B80                                           ;  line (top command line) on the
```

```
                                  0B80                                              ;  terminal.
        00002260'EF   02     C3   0B80          SUBL3   #2,LAST_LINE,-              ; Save the number of the last line on
              00002258'EF         0B87                  LAST_DATA_LINE              ;  the terminal to be used to display
                                  0B8C                                              ;  data.
                                  0B8C
                                  0B8C ;
                                  0B8C ; Save local system ID in LOCSYSID.
                                  0B8C ;
                                  0B8C
00000005'EF   00000000'GF    D0   0B8C 63$:     MOVL    G^SCS$GB_SYSTEMID,LOCSYSID
00000009'EF   00000004'GF    B0   0B97          MOVW    G^SCS$GB_SYSTEMID+4,LOCSYSID+4
                                  0BA2
                                  0BA2 ;
                                  0BA2 ; Get the local node name.
                                  0BA2 ;
                                  0BA2
                                  0BA2          $CMKRNL_S -
                                  0BA2                  ROUTIN=GET_NODE_NAME
                                  0BB1
                                  0BB1 ;
                                  0BB1 ; Process command line qualifier /REPORT=.
                                  0BB1 ;
              0000001B'EF    DF   0BB1          PUSHAL  REPORT_QUAL                 ; Was the report name specified?
        00000000'GF   01     FB   0BB7          CALLS   #1,G^CLI$PRESENT            ;
                                  0BBE          JLBC    R0,80$                      ; If not default to cluster report.
              00001DA3'EF    DF   0BC4          PUSHAL  GET_VAL_DSCR                ; Get the report name.
              0000001B'EF    DF   0BCA          PUSHAL  REPORT_QUAL                 ;
        00000000'GF   02     FB   0BD0          CALLS   #2,G^CLI$GET_VALUE          ;
                                  0BD7          JLBC    R0,80$                      ; Default to cluster report on error.
      4C 8F   00001DB3'EF    91   0BDD          CMPB    RET_BUF,#^A/L/              ; Local report requested?
                      25     13   0BE5          BEQL    75$                         ; Yes go setup for the local report.
    4C43 8F   00001DB3'EF    B1   0BE7          CMPW    RET_BUF,#^A/CL/             ; Cluster report requested?
                      61     13   0BF0          BEQL    80$                         ; Yes, go setup for the cluster report.
    5943 8F   00001DB3'EF    B1   0BF2          CMPW    RET_BUF,#^A/CY/             ; /REPORT=CYCLE specified?
                      09     12   0BFB          BNEQ    72$                         ; Branch if not.
                      30     C8   0BFD          BISL    #MF$M_CONT!MF$M_CYCLE,-     ; Set cycle and continuous display
              00000476'EF         0BFF                  MISC_FLAGS                  ;  flags.
                      4D     11   0C04          BRB     80$                         ; Set up to start cycling with the
                                  0C06                                              ;  cluster report.
              50      14     D0   0C06 72$:     MOVL    #SS$_BADPARAM,R0            ; Exit status is bad parameter value.
                    0336     31   0C09          BRW     MAIN_ERR                    ; Exit.
                                  0C0C
                                  0C0C
                                  0C0C ;
                                  0C0C ; Setup for the local report.
                                  0C0C ;
                                  0C0C
              00370006 8F    D0   0C0C 75$:     MOVL    #CLS$M_ALL_SYS!CLS$M_CSB,-
              00002204'EF         0C12                  CLUST_SAVE
00000001'EF   00000180 8F    D0   0C17          MOVL    #CLS$M_LPRT!CLS$M_ERR,CLASS_SELECT
                      01     8A   0C22          BICB    #RPT$M_CLSTR,-              ; Set current report to local.
              00000000'EF         0C24                  CURRENT_REPORT
00002274'EF   00000000'EF    DE   0C29          MOVAL   LOCAL_KEYTAB,KEYTAB_ADRS; Store address of local status keyword
                                  0C34                                          ;  table.
00002264'EF   00000000'EF    DE   0C34          MOVAL   LOCAL_TABLE,ADRS_TABLE  ; Store address of local address table.
00002268'EF   00000000'EF    DE   0C3F          MOVAL   LOCAL_BITMAP,BITMAP_ADRS; Store address of local bitmap.
        0000226C'EF   00     D0   0C4A          MOVL    #LOCAL_MAP_LEN,MAP_LEN  ; Store length of local bitmap.
```

```
                         45   11  0C51          BRB     85$                     ; Branch to common code.
                                  0C53
                                  0C53   ;
                                  0C53   ; Setup for the cluster report.
                                  0C53   ;
                                  0C53
                   00370006 8F    D0  0C53   80$:   MOVL    #CLS$M_ALL_SYS!CLS$M_CSB,-
                   00000001'EF        0C59                  CLASS_SELECT
                   00000180 8F    D0  0C5E          MOVL    #CLS$M_LPRT!CLS$M_ERR,-
                   00002200'EF        0C64                  LOCAL_SAVE
                         01       88  0C69          BISB    #RPT$M_CLSTR,-          ; Set current report to clusters.
                   00000000'EF        0C6B                  CURRENT_REPORT          ;
                   00000000'EF    DE  0C70          MOVAL   CLUSTER_KEYTAB,-        ; Store address of cluster keyword
                   00002274'EF        0C76                  KEYTAB_ADRS             ;  table.
      00002264'EF  00000000'EF    DE  0C7B          MOVAL   CLUSTER_TABLE,ADRS_TABLE; Store address of cluster table.
                   00000000'EF    DE  0C86          MOVAL   CLUSTER_BITMAP,-        ; Store address of cluster bitmap.
                   00002268'EF        0C8C                  BITMAP_ADRS             ;
              0000226C'EF    02   D0  0C91          MOVL    #CLUSTER_MAP_LEN,MAP_LEN; Store length of cluster bitmap.
                                      0C98
                                      0C98   ;
                                      0C98   ; Check for /CONTINUOUS.
                                      0C98   ;
                                      0C98
                   00000029'EF    DF  0C98   85$:   PUSHAL  CONTINUOUS_QUAL         ; Was /CONTINUOUS specified?
              00000000'GF    01   FB  0C9E          CALLS   #1,G^CLI$PRESENT        ;
                         07  50   E9  0CA5          BLBC    R0,87$                  ; Branch if not.
              00000476'EF    10   C8  0CA8          BISL    #MF$M_CONT,MISC_FLAGS   ; Set continuous display flag.
                                      0CAF
                                      0CAF   ;
                                      0CAF   ; Check for /INTERVAL.
                                      0CAF   ;
                                      0CAF
                   0000003B'EF    DF  0CAF   87$:   PUSHAL  INTERVAL_QUAL           ; Was /INTERVAL specified?
              00000000'GF    01   FB  0CB5          CALLS   #1,G^CLI$PRESENT        ;
                         56  50   E9  0CBC          BLBC    R0,88$                  ; Branch if not.
              00000476'EF    10   C8  0CBF          BISL    #MF$M_CONT,MISC_FLAGS   ; Set continuous display flag.
                   00001DAB'EF    DF  0CC6          PUSHAL  RET_BUF_DSCR            ; Get the time string.
                   00001DA3'EF    DF  0CCC          PUSHAL  GET_VAL_DSCR            ;
                   0000003B'EF    DF  0CD2          PUSHAL  INTERVAL_QUAL           ;
              00000000'GF    03   FB  0CD8          CALLS   #3,G^CLI$GET_VALUE      ;
                                      0CDF          JLBC    R0,99$                  ; Branch on error.
                   000021AC'EF    DF  0CE5          PUSHAL  DELAY                   ; Convert the desired number of seconds
                   00001DB3'EF    DF  0CEB          PUSHAL  RET_BUF                 ;  to a longword.
                   00001DAB'EF    DD  0CF1          PUSHL   RET_BUF_DSCR            ;
              00000000'GF    03   FB  0CF7          CALLS   #3,G^LIB$CVT_DTB        ;
                                      0CFE          JLBC    R0,99$                  ; Branch on error.
00    000021AC'EF  FF676980 8F    7A  0D04          EMUL    #-10000000,DELAY,#0,-   ; Convert the number of seconds to
                   000021AC'EF        0D10                  DELAY                   ;  binary time.
                                      0D15
                   000019DC'EF    16  0D15   88$:   JSB     SET_REGION_DATA         ; Set up scrolling region.
                                      0D1B
                                      0D1B   ;
                                      0D1B   ; Save a copy of the cluster bitmap for INITIALIZE command.
                                      0D1B   ;
                                      0D1B
                         55  02   D0  0D1B   90$:   MOVL    #CLUSTER_MAP_LEN,R5
            57     00000000'EF    DE  0D1E          MOVAL   CLUSTER_BITMAP,R7
```

```
      56  0000000C'EF   DE  0D25             MOVAL   CL_SAVE_BITMAP,R6
             86    87   D0  0D2C  91$:       MOVL    (R7)+,(R6)+
                FA 55   F4  0D2F             SOBGEQ  R5,91$
                            0D32
                            0D32  ;
                            0D32  ; Save a copy of the local ports bitmap for INITIALIZE command.
                            0D32  ;
             55    00   D0  0D32             MOVL    #LOCAL_MAP_LEN,R5
      57  00000000'EF   DE  0D35             MOVAL   LOCAL_BITMAP,R7
      56  00000004'EF   DE  0D3C             MOVAL   LOC_SAVE_BITMAP,R6
             86    87   D0  0D43  92$:       MOVL    (R7)+,(R6)+
                FA 55   F4  0D46             SOBGEQ  R5,92$
                            0D49
                            0D49
                            0D49  ;
                            0D49  ; Set the initial number of system blocks and connections that can be handled
                            0D49  ; by the snapshot routine.  These numbers will be used in the MAIN_LOOP routine
                            0D49  ; to determine the size of buffers to allocate for system blocks and
                            0D49  ; connections.  The size of buffers for circuits and cluster system blocks will
                            0D49  ; be derived from the number of system blocks.
                            0D49  ;
                            0D49  ; The initial number of connections is set to the number of connections allowed
                            0D49  ; by the SYSGEN SCSCONNCNT parameter but not to exceed the parameter default
                            0D49  ; (DEF_CONNCNT).  This is to protect from the possibility of the parameter
                            0D49  ; being set much higher than necessary. (maximum is 32k)
                            0D49  ;
                            0D49  ; The initial number of system blocks is set to 12 which is in affect
                            0D49  ; the same as setting it to 16 because it will be increased by 4 on the first
                            0D49  ; pass to allow for expansion.
                            0D49  ;
         00000000'GF    B1  0D49             CMPW    G^SCS$GW_CDTCNT,-           ; Are there less connections allowed
                  12        0D4F                     #DEF_CONNCNT                ;  on this system then there normally
                            0D50                                                 ;  are by default?
                  09    1F  0D50             BLSSU   93$                         ; Branch if so.
     00000041'EF  12    B0  0D52             MOVW    #DEF_CONNCNT,CDT_DSCR+2     ; Otherwise take the default.
                  0B    11  0D59             BRB     94$                         ; Jump to common code.
         00000000'GF    B0  0D5B  93$:       MOVW    G^SCS$GW_CDTCNT,-           ; Store the actual number allowed.
         00000041'EF        0D61                     CDT_DSCR+2                  ;
     00000029'EF  0C    B0  0D66  94$:       MOVW    #12,SB_DSCR+2               ; Fake the number of known system
                            0D6D                                                 ;  blocks.  This will be expanded to
                            0D6D                                                 ;  16 on the first pass.
                            0D6D
                            0D6D  ;
                            0D6D  ; Clear the terminals screen.
                            0D6D  ;
                1660    30  0D6D             BSBW    CLEAR_SCREEN                ; Clear the screen.
                            0D70             JLBC    R0,99$                      ; Branch on error.
                            0D76
                            0D76  ;
                            0D76  ; Translate SHOW_CLUSTER$INIT.  If the logical name is defined then go execute
                            0D76  ; the initialization file.
                            0D76  ;
                            0D76
                            0D76             $TRNLOG_S -                         ; Translate the logical name
                            0D76                     LOGNAM=INIT_NAME,-          ;  SHOW_CLUSTER$INIT.
                            0D76                     RSLLEN=TRAN_DSCR,-          ;
                            0D76                     RSLBUF=TRAN_DSCR            ;
```

```
        50  00000629 8F  D1  0D95  3944          CMPL    #SS$_NOTRAN,R0            ; Was SHOW_CLUSTER$INIT defined?
                     31  13  0D9C  3945          BEQL    95$                       ; Branch if not.
            00000001'EF  DE  0D9E  3946          MOVAL   DEF_EXT,-                 ; Store default file extention in FAB.
            0000004C'EF      0DA4  3947                  INFAB+FAB$L_DNA           ;
                     04  90  0DA9  3948          MOVB    #DEF_EXT_LEN,-            ; Store length of default extention in
            00000051'EF      0DAB  3949                  INFAB+FAB$B_DNS           ;  FAB.
         00001215'EF  00  FB  0DB0  3950          CALLS   #0,EXECUTE                ; Execute initialization file.
                             0DB7  3951          JLBC    R0,99$                    ; Branch on error.
            00000005'EF  DE  0DBD  3952          MOVAL   DEF_NAME2,-               ; Store default filename in FAB.
            0000004C'EF      0DC3  3953                  INFAB+FAB$L_DNA           ;
                     16  90  0DC8  3954          MOVB    #DEF_NAME2_LEN,-          ; Store length of default filename in
            00000051'EF      0DCA  3955                  INFAB+FAB$B_DNS           ;  FAB.
                             0DCF  3956
            00000059'EF  DF  0DCF  3957 95$:     PUSHAL  ENDING_QUAL               ; Was /ENDING specified?
         00000000'GF  01  FB  0DD5  3958          CALLS   #1,G^CLI$PRESENT          ;
                  55 50  E9  0DDC  3959          BLBC    R0,96$                    ; Branch if not.
            00001DAB'EF  DF  0DDF  3960          PUSHAL  RET_BUF_DSCR              ; Get the time string.
            00001DA3'EF  DF  0DE5  3961          PUSHAL  GET_VAL_DSCR              ;
            00000059'EF  DF  0DEB  3962          PUSHAL  ENDING_QUAL               ;
         00000000'GF  03  FB  0DF1  3963          CALLS   #3,G^CLI$GET_VALUE        ;
                             0DF8  3964          JLBC    R0,99$                    ; Branch on error.
                             0DFE  3965
            000021B4'EF  7F  0DFE  3966          PUSHAQ  A_TIME                    ; Convert the time string to binary.
            00001DAB'EF  DF  0E04  3967          PUSHAL  RET_BUF_DSCR              ;
         00000000'GF  02  FB  0E0A  3968          CALLS   #2,G^LIB$CVT_TIME         ;
                             0E11  3969          JLBC    R0,99$                    ; Branch on error.
                             0E17  3970
                             0E17  3971          $SETIMR_S -                       ; Set the timer to force a control-C
                             0E17  3972                  DAYTIM = A_TIME,-         ;  when it expires.
                             0E17  3973                  ASTADR = CNTRLC_AST_DEL   ;
                             0E2E  3974          JLBC    R0,99$                    ; Branch on error.
                             0E34  3975
            00000067'EF  DF  0E34  3976 96$:     PUSHAL  BEGIN_QUAL                ; Was /BEGINNING specified?
         00000000'GF  01  FB  0E3A  3977          CALLS   #1,G^CLI$PRESENT          ;
                  05 50  E8  0E41  3978          BLBS    R0,97$                    ; Branch if so.
                  50 01  D0  0E44  3979          MOVL    #SS$_NORMAL,R0            ; Otherwise, make sure we don't
                             0E47  3980                                            ;  consider the error fatal.
                     79  11  0E47  3981          BRB     975$                      ; Branch to common.
            00001DAB'EF  DF  0E49  3982 97$:     PUSHAL  RET_BUF_DSCR              ; Get the time string.
            00001DA3'EF  DF  0E4F  3983          PUSHAL  GET_VAL_DSCR              ;
            00000067'EF  DF  0E55  3984          PUSHAL  BEGIN_QUAL                ;
         00000000'GF  03  FB  0E5B  3985          CALLS   #3,G^CLI$GET_VALUE        ;
                             0E62  3986          JLBC    R0,99$                    ; Branch on error.
                             0E68  3987
            000021B4'EF  7F  0E68  3988          PUSHAQ  A_TIME                    ; Convert the time string to binary.
            00001DAB'EF  DF  0E6E  3989          PUSHAL  RET_BUF_DSCR              ;
         00000000'GF  02  FB  0E74  3990          CALLS   #2,G^LIB$CVT_TIME         ;
                             0E7B  3991          JLBC    R0,99$                    ; Branch on error.
                             0E81  3992
                             0E81  3993          $SCHDWK_S -                       ; Leave a wakeup call.
                             0E81  3994                  DAYTIM = A_TIME           ;
                  7F 50  E9  0E94  3995          BLBC    R0,99$                    ; Branch on error.
                             0E97  3996
            00001DAB'EF  DE  0E97  3997          MOVAL   RET_BUF_DSCR,-            ; Save address of time string in error
            000004A9'EF      0E9D  3998                  ERR_MSGVEC+12             ;  message vector.
 000004B9'EF  00000000'8F  D0  0EA2  3999          MOVL    #SHOW$_HIBER,CMD_STATUS   ; Tell the user we're going into
                   06AD  30  0EAD  4000          BSBW    SIGNAL                    ;  hibernation.
```

```
                              DE  0EB0          MOVAL   <TPARSE_BLK+-             ; Restore error message vector to
                                  0EB1                  TPA$L_STRINGCNT>,-        ;  normal.
  000004A9'EF   000004CF'EF       0EB1                  ERR_MSGVEC+12             ;
                                  0EBB
                                  0EBB          $HIBER_S                          ; Off to sleep...
                                  0EC2
        03 00002246'EF    05  E1  0EC2 975$:    BBC     #DVT$V_DISK,IN_DEV,98$    ; Branch if input isn't from a file.
                        0049  31  0ECA          BRW     99$                       ; And finish up.
                                  0ECD
                                  0ECD ;
                                  0ECD ; Set up control-c handler.
                                  0ECD ;
                                  0ECD
                                  0ECD 98$:     $QIOW_S -                         ; Setup a control-C ast handler.
                                  0ECD                  CHAN=IN_CHAN,-            ;
                                  0ECD                  FUNC=#IO$_SETMODE!IO$M_CTRLCAST,-
                                  0ECD                  P1=CNTRLC_AST_DEL         ;
                                  0EF2
  0000054F'EF   00002222'EF   3C  0EF2          MOVZWL  SCREEN_WIDTH,VALUE        ; Save number of columns for set screen
          00000FBA'EF     00  FB  0EFD          CALLS   #0,SET_SCREEN             ; Set the screen width to 80 or 132
                                  0F04                                            ;  column mode.
                       0F 50  E9  0F04          BLBC    R0,99$                    ; Branch on error.
                                  0F07
                                  0F07 ;
                                  0F07 ; Set up for first AST.
                                  0F07 ;
                                  0F07
                                  0F07          $CLREF_S -                        ; Clear ast event flag.
                                  0F07                  EFN=#EVF$V_AST_FLAG
                       03 50  E9  0F10          BLBC    R0,99$                    ; Branch on error.
                        0B23  30  0F13          BSBW    SET_AST                   ; Set first AST for unsolicited data.
                                  0F16
                     00FC 8F  BA  0F16 99$:     POPR    #^M<R2,R3,R4,R5,R6,R7>
                              05  0F1A          RSB                               ; Return with status.
                                  0F1B
                                  0F1B
```

```
                          0F1B  ;++
                          0F1B  ;
                          0F1B  ; FUNCTIONAL DESCRIPTION:
                          0F1B  ;
                          0F1B  ; This routine simply gets the node name of the local system from the local
                          0F1B  ; system block and stores it in the NODE_NAME_DSCR.  The nodename is then used
                          0F1B  ; by the GET_HEADER routine to display the nodename in the header.
                          0F1B  ;
                          0F1B  ; CALLING SEQUENCE:
                          0F1B  ;
                          0F1B  ;         CALLS   #0,GET_NODE_NAME
                          0F1B  ;
                          0F1B  ; CALLED BY:
                          0F1B  ;
                          0F1B  ;         INIT
                          0F1B  ;
                          0F1B  ; INPUTS:
                          0F1B  ;
                          0F1B  ;         Nodename stored in the local system block.
                          0F1B  ;
                          0F1B  ; OUTPUTS:
                          0F1B  ;
                          0F1B  ;         Nodename stored in NODE_NAME_DSCR.
                          0F1B  ;
                          0F1B  ; RETURN STATUS:
                          0F1B  ;
                          0F1B  ;         None.
                          0F1B  ;
                          0F1B  ;--
                          0F1B
                  003C    0F1B  .ENTRY  GET_NODE_NAME,^M<R2,R3,R4,R5>
                          0F1D
                          0F1D  ;
                          0F1D  ; Get the number of bytes in the nodename.
                          0F1D  ;
00000044'GF        90     0F1D          MOVB    G^SCS$GA_LOCALSB+SB$T_NODENAME,-
00000448'EF               0F23                  NODE_NAME_DSCR
                          0F28  ;
                          0F28  ; Store the node name in the node name descriptor.
                          0F28  ;
00000448'EF        28     0F28          MOVC3   NODE_NAME_DSCR,-
00000045'GF               0F2E                  G^SCS$GA_LOCALSB+SB$T_NODENAME+1,-
00000450'EF               0F33                  NODE_NAME
                          0F38
                   04     0F38          RET
                          0F39
                          0F39
```

```
                                  0F39      .SBTTL  Exit handler.
                                  0F39 ;
                                  0F39 ;        This is the entry point for the exit routine established during
                                  0F39 ;        initialization.  It simply moves the status stored in STATUS by
                                  0F39 ;        the $EXIT_S to R0 and falls through to the MAIN_ERR routine.
                                  0F39 ;
                                  0F39
                          0000    0F39 .ENTRY   EXIT_RTN,^M<>
                                  0F3B
        50   00000491'EF    D0    0F3B          MOVL    STATUS,R0                  ; Get the exit status.
                                  0F42
                                  0F42 ;++
                                  0F42 ;
                                  0F42 ; FUNCTIONAL DESCRIPTION:
                                  0F42 ;
                                  0F42 ;        This routine handles all cleanup before the program exits.  It restores
                                  0F42 ;        the original terminal characteristics, restores the original screen
                                  0F42 ;        width, and deassigns the channels assigned to the terminal and mailbox.
                                  0F42 ;        An error encountered while trying to do the cleanup is considered fatal
                                  0F42 ;        and the image immediately exits.
                                  0F42 ;
                                  0F42 ; INPUTS:
                                  0F42 ;
                                  0F42 ;        Error status in R0.
                                  0F42 ;
                                  0F42 ; IMPLICIT OUTPUTS:
                                  0F42 ;
                                  0F42 ;        Error message output to terminal if status in R0 is not success.
                                  0F42 ;
                                  0F42 ; RETURN STATUS:
                                  0F42 ;
                                  0F42 ;        Error status in R0.
                                  0F42 ;
                                  0F42 ;--
                                  0F42
                                  0F42 MAIN_ERR:
                     02     C8    0F42          BISL    #MFSM_EXIT_FLAG,-          ; Indicate exit in progress.
               00000476'EF        0F44                  MISC_FLAGS                 ;
        00002247'EF  05     E1    0F49          BBC     #DVT$V_DISK,OUT_DEV,-      ; Branch if I/O device is a terminal.
                     03           0F50                  5$                         ;
                   0094     31    0F51          BRW     MAIN_EXIT                  ; Otherwise, skip the following.
        000004A1'EF  50     D0    0F54 5$:      MOVL    R0,ERR_MSGVEC+4            ; Save status for error message.
                                  0F5B
                                  0F5B          $QIOW_S -                          ; Make sure graphics mode is off.
                                  0F5B                  CHAN=OUT_CHAN,-
                                  0F5B                  FUNC=#IO$_WRITEVBLK,-
                                  0F5B                  P1=#^A//,-
                                  0F5B                  P2=#1
                                  0F80          JLBC    R0,MAIN_EXIT               ; Branch on error.
                                  0F86
0000054F'EF  00002216'EF    3C    0F86          MOVZWL  SAVE_CHAR_BUF+2,VALUE      ; Get the original screen width.
        00000FBA'EF  00     FB    0F91          CALLS   #0,SET_SCREEN              ; Reset the screen width.
                                  0F98          JLBC    R0,MAIN_EXIT               ; Branch on error.
                                  0F9E
                   0A2D     30    0F9E          BSBW    SET_REGION_FULL            ; Reset the scrolling region to use the
                                  0FA1                                             ;   full screen. (Added for foreign
                                  0FA1                                             ;   terminals emulating VT100s).
```

```
                            0FA1
      13 000004A1'EF    E9  0FA1        BLBC    ERR_MSGVEC+4,10$            ; Don't clear the screen if exiting
                            0FA8                                            ;  with error status.
 00000476'EF      04    E1  0FA8        BBC     #MF$V_CONT,MISC_FLAGS,-     ; Don't clear the screen if display
                  0B        0FAF                10$                         ;  is not in continuous update mode.
                141D    30  0FB0        BSBW    CLEAR_SCREEN                ; Clear the screen.
                            0FB3        JLBC    R0,MAIN_EXIT                ; Branch on error.
                  00    11  0FB9        BRB     10$                         ; Exit status is normal - don't output
                            0FBB                                            ;  any message.
                            0FBB
                  07    E1  0FBB 10$:   BBC     #MF$V_NEW_CHAR,-            ; Branch if the terminal characteristics
      25 00000476'EF        0FBD                MISC_FLAGS,MAIN_EXIT        ;  were never changed.
                            0FC3        $QIOW_S -                           ; Restore old terminal characteristics.
                            0FC3                CHAN=OUT_CHAN,-             ;
                            0FC3                FUNC=#IO$_SETMODE,-         ;
                            0FC3                P1=SAVE_CHAR_BUF,-          ;
                            0FC3                P2=#CHAR_BUF_LEN            ;
                            0FE8
                            0FE8 MAIN_EXIT:
 000004A1'EF      01    D1  0FE8        CMPL    #SS$_NORMAL,ERR_MSGVEC+4; Is the original exit status success?
                  07    13  0FEF        BEQL    99$                         ; If so exit with status from MAIN_ERR
                            0FF1                                            ;  routine.
 50    000004A1'EF      D0  0FF1        MOVL    ERR_MSGVEC+4,R0             ; Otherwise get the original status
                            0FF8                                            ;  back.
                        04  0FF8 99$:   RET                                 ; And exit...
                            0FF9
                            0FF9
```

```
                                  0FF9  ;++
                                  0FF9  ;
                                  0FF9  ; FUNCTIONAL DESCRIPTION:
                                  0FF9  ;
                                  0FF9  ;         This routine coordinates the generation of each report by calling
                                  0FF9  ;         the appropriate routines based on the classes selected by the
                                  0FF9  ;         user.
                                  0FF9  ;
                                  0FF9  ;--
                                  0FF9
                                  0FF9  DISPLAY:
               0480 8F      BB    0FF9          PUSHR   #^M<R7,R10>
       00002248'EF   01     D0    0FFD          MOVL    #1,LINE_NUM                     ; Initialize line number.
    1F 00002247'EF   01     E1    1004          BBC     #DVT$V_DCRT,OUT_DEV,10$         ; Branch if output device isn't a
                                  100C                                                  ;  DECcrt.
            0000224C'EF     DF    100C          PUSHAL  COL_NUM                         ; Make sure line drawing and special
            00002248'EF     DF    1012          PUSHAL  LINE_NUM                        ;  graphics character set in G1 area in
            00000168'EF     DF    1018          PUSHAL  GRAPHICS_DSGNTR                 ;  DECcrt on each screen update.
       00000000'GF   03     FB    101E          CALLS   #3,G^LIB$PUT_SCREEN             ;
                                  1025          JLBC    R0,DISPLAY_ERROR                ; Branch on error.
                                  102B
                     08     8A    102B  10$:    BICB    #SCR$M_MORE_DWN,-               ; Clear flag that indicates there is
            0000047A'EF           102D                  SCROLL_FLAGS                    ;  data off the bottom of the screen.
00000438'EF    00000434'EF  D0    1032          MOVL    LINE_DIF,PRIOR_LINE_DIF         ; Save record of lines changed on the
                                  103D                                                  ;  last pass.
00000434'EF    00000444'EF  D0    103D          MOVL    SCROLL_DIF,LINE_DIF             ; Clear record of lines changed this
                                  1048                                                  ;  pass, but set the bits for any
                                  1048                                                  ;  lines that have to be redrawn
                                  1048                                                  ;  because of scrolling.
            0000043C'EF     D0    1048          MOVL    BLANK_LINES,-                   ; Save record of blank lines on the
            00000440'EF           104E                  PRIOR_BLANK_LINES               ;  last pass.
            0000043C'EF     D4    1053          CLRL    BLANK_LINES                     ; Clear record of blank lines this
                                  1059                                                  ;  pass.
    18 00000476'EF   03     E1    1059          BBC     #MFS$V_RIP,MISC_FLAGS,20$       ; Branch if refresh not in progress.
            000021C8'EF     D4    1061          CLRL    CLUB_HEADER_SIZE                ; Initialize CLUB header size...
            000021CC'EF     D4    1067          CLRL    SCS_HEADER_SIZE                 ; ...SCS header size...
            00000427'EF     94    106D          CLRB    CLUB_NUM_FIELDS                 ; ...number of CLUB fields...
            00000426'EF     94    1073          CLRB    SCS_NUM_FIELDS                  ; ...number of SCS fields...
                                  1079
                                  1079  ;
                                  1079  ; Go allocate some buffer space, lock in memory and take a snapshot of the
                                  1079  ; SCS and port databases.
                                  1079  ;
                                  1079
                  F4B1 CF   16    1079  20$:    JSB     DO_SNAPSHOT                     ;
                                  107D          JLBC    R0,DISPLAY_ERROR                ; Branch on error.
                                  1083          JBC     #RPT$V_CLSTR,-                  ; Branch if we're not displaying the
                                  1083                  CURRENT_REPORT,-                ;  cluster report.
                                  1083                  DISPLAY_PTP                     ;
            0000007F 8F     D3    108E          BITL    #CLS$M_ANY_CLUS,-               ; Any Cluster report information to be
            00000001'EF           1094                  CLASS_SELECT                    ;  displayed?
                     03     12    1099          BNEQ    30$                             ; Branch if so.
                   01A7     31    109B          BRW     DISPLAY_END                     ; And go finish up.
                     00     E1    109E  30$:    BBC     #CLS$V_CLUB,-                   ; Branch if display of CLUB is disabled.
    63 00000001'EF                10A0                  CLASS_SELECT,DISPLAY_SCS;
                                  10A6  ;
                                  10A6  ; Display the CLUSTER class at the top of the screen.  This class
```

```
                               10A6  ; contains general cluster information from the CLUB.
                               10A6  ;
                               10A6
                               10A6  DISPLAY_CLUB:
        00000001'EF     D0     10A6          MOVL    CLASS_SELECT,-            ; Save mask of selected classes.
        000021FC'EF            10AC                  SAVED_CLASSES             ;
        0000007E 8F     CA     10B1          BICL    #CLS$M_ANY_SCS,-          ; Temporarily disable all classes but
        00000001'EF            10B7                  CLASS_SELECT              ;  the CLUB.
    000021F8'EF    01   C8     10BC          BISL    #CLS$M_CLUB,DATA_CNTRL    ; Enable display of fields in the CLUB.
                   01   D0     10C3          MOVL    #CLS$M_CLUB,-             ; Set current class to CLUSTER.
        000021F4'EF            10C5                  CURRENT_CLASS             ;
               0B79     30     10CA          BSBW    GET_HEADER                ; Output CLUB header to screen buffer.
                               10CD          JLBC    R0,DISPLAY_ERROR          ; Branch on error.
        000021C8'EF     D0     10D3          MOVL    CLUB_HEADER_SIZE,-        ; If no refresh and displaying both CLUB
        000021C4'EF            10D9                  HEADER_SIZE               ;  and SCS, the HEADER_SIZE will be set
                               10DE                                            ;  to the SCS header size, so we must
                               10DE                                            ;  get the real CLUB_HEADER_SIZE back.
        0000004B'EF     DF     10DE          PUSHAL  CLUB_BUF                  ; Push address of the CLUB.
    00002729'EF    01   FB     10E4          CALLS   #1,GET_DATA               ; Get a line of data.
                               10EB  ;
                               10EB  ; Since the number of lines in the CLUB portion of the display will never
                               10EB  ; change, we don't have to call DISPLAY_COMMON.  Instead we output the
                               10EB  ; bottom line of the display box only when doing a refresh pass.
                               10EB  ;
 05 00000476'EF    03   E1     10EB          BBC     #MF$V_RIP,MISC_FLAGS,20$; Branch if refresh not in progress.
               0FFB     30     10F3          BSBW    DRAW_LAST_LINE            ; Draw bottom line of CLUB display box.
                 06     11     10F6          BRB     30$                       ; Branch to common code.
        00002248'EF     D6     10F8  20$:    INCL    LINE_NUM                  ; Skip over bottom line of CLUB display
                               10FE                                            ;  box.
        000021FC'EF     D0     10FE  30$:    MOVL    SAVED_CLASSES,-           ; Restore mask of selected classes.
        00000001'EF            1104                  CLASS_SELECT              ;
                               1109
                               1109  ;
                               1109  ; Display any SCS data (SYSTEMS, CIRCUITS, CONNECTION, COUNTERS,
                               1109  ; or CREDITS classes) or any CSB data (MEMBERS class).
                               1109  ;
                               1109
                               1109  DISPLAY_SCS:
        0000007E 8F     D3     1109          BITL    #CLS$M_ANY_SCS,-          ; Any SCS or Cluster System Block info
        00000001'EF            110F                  CLASS_SELECT              ;  to be displayed?
                               1114          JEQL    DISPLAY_END               ; Branch if not.
        00000001'EF     D0     1119          MOVL    CLASS_SELECT,-            ; Save mask of selected classes.
        000021FC'EF            111F                  SAVED_CLASSES             ;
00000460'EF   0000045C'EF  D0  1124          MOVL    V_SCROLL,V_SCROLL_LINE    ; Set the number of data lines to skip.
              F7C9'     30     112F          BSBW    GET_NUM_PB                ; Get the number of circuits on each
                               1132                                            ;  system.
    00000001'EF    01   CA     1132          BICL    #CLS$M_CLUB,CLASS_SELECT; We've already displayed CLUB info.
    000021F4'EF    02   D0     1139          MOVL    #CLS$M_SYS,CURRENT_CLASS; Set current class to SYSTEMS.
               0B03     30     1140          BSBW    GET_HEADER                ; Output SCS header to screen buffer.
                               1143          JLBC    R0,DISPLAY_ERROR          ; Branch on error.
        000021CC'EF     D0     1149          MOVL    SCS_HEADER_SIZE,-         ; If no refresh, and displaying both
        000021C4'EF            114F                  HEADER_SIZE               ;  CLUB and SCS, the header size will
                               1154                                            ;  be set to the CLUB header size.  We
                               1154                                            ;  must get the SCS header size back.
                 0C     13     1154          BEQL    10$                       ; Branch if nothing to display.
               015C     30     1156          BSBW    SCS_REPORT                ; Output the SCS data.
               00FE     30     1159          BSBW    DISPLAY_COMMON            ; Execute common display code.
```

```
                         115C        JLBC    R0,DISPLAY_ERROR          ; Branch on error.
        000021FC'EF  D0  1162  10$:  MOVL    SAVED_CLASSES,-           ; Restore mask of selected classes.
        00000001'EF      1168                CLASS_SELECT              ;
               00D5  31  116D        BRW     DISPLAY_END               ; Finish up display.
                         1170
                         1170
                         1170  ;
                         1170  ; Display the Port-to-Port data.  (LOCAL_PORTS and ERRORS classes).
                         1170  ;
                         1170
                         1170  DISPLAY_PTP:
        00000460'EF  D4  1170        CLRL    V_SCROLL_LINE             ; Insure that vertical scrolling
                         1176                                          ;  doesn't occur in the local report.
        00000080 8F  D0  1176        MOVL    #CLS$M_LPRT,-             ; Set current class to LOCAL_PORTS.
        000021F4'EF      117C                CURRENT_CLASS             ;
               0AC2  30  1181        BSBW    GET_HEADER                ; Output SCS header to screen buffer.
                         1184        JLBC    R0,DISPLAY_ERROR          ; Branch on error.
               0386  30  118A        BSBW    LOCAL_REPORT              ; Output the PTP data.
                         118D        JLBC    R0,DISPLAY_ERROR          ; Branch on error.
               00C4  30  1193        BSBW    DISPLAY_COMMON            ; Execute common display code.
                         1196        JLBC    R0,DISPLAY_ERROR          ; Branch on error.
                         119C  ;
                         119C  ; For the Local_Ports report, we also output the number of buffer descriptors
                         119C  ; free and in use below the rest of the display.
                         119C  ;
               125E  30  119C        BSBW    ERASE_TWO_LINES           ; Leave a couple of blank lines.
                         119F        JLBC    R0,DISPLAY_ERROR          ; Branch on error.
    5A  00000023'EF  DE  11A5        MOVAL   NUM_BUF_DSCR_DSCR,R10     ; Point to descriptor for the number of
                         11AC                                          ; buffer descriptors in use.
          57   10 AA D0  11AC        MOVL    FD$L_OFFSET(R10),R7       ; Get the SCS offset from the
                         11B0                                          ;  descriptor.
               13CC  30  11B0        BSBW    FORMAT_DATA               ; Format the data.
                         11B3        JLBC    R0,DISPLAY_ERROR          ; Branch on error.
        0000224C'EF  3F  11B9        PUSHAW  COL_NUM                   ; Output the data to the report.
        00002248'EF  3F  11BF        PUSHAW  LINE_NUM                  ;
        00002153'EF  DF  11C5        PUSHAL  FORMAT_BUF_DSCR           ;
     00000000'GF  03 FB  11CB        CALLS   #3,G^LIB$PUT_SCREEN       ;
                         11D2        JLBC    R0,DISPLAY_ERROR          ; Branch on error.
        000001DF'EF  DF  11D8        PUSHAL  BUFF_MSG                  ; Print "CI buffer descriptors in use."
     00000000'GF  01 FB  11DE        CALLS   #1,G^LIB$PUT_SCREEN       ;
              6D 50  E9  11E5        BLBC    R0,DISPLAY_ERROR          ; Branch on error.
     00000000'GF  00 FB  11E8        CALLS   #0,G^LIB$ERASE_LINE       ; Erase the rest of the line.
              63 50  E9  11EF        BLBC    R0,DISPLAY_ERROR          ; Branch on error.
        00002248'EF  D6  11F2        INCL    LINE_NUM                  ; Free buffer descriptors on next line.
        0000004C'EF  DE  11F8        MOVAL   NUM_FREE_BUF_DSCR_DSCR,-  ; Point to descriptor for number of
                 5A      11FE                R10                       ;  buffer descriptors free.
          57   10 AA D0  11FF        MOVL    FD$L_OFFSET(R10),R7       ; Get the SCS offset from the
                         1203                                          ;  descriptor.
               1379  30  1203        BSBW    FORMAT_DATA               ; Format the data.
              4C 50  E9  1206        BLBC    R0,DISPLAY_ERROR          ; Branch on error.
        0000224C'EF  3F  1209        PUSHAW  COL_NUM                   ; Output the data to the report.
        00002248'EF  3F  120F        PUSHAW  LINE_NUM                  ;
        00002153'EF  DF  1215        PUSHAL  FORMAT_BUF_DSCR           ;
     00000000'GF  03 FB  121B        CALLS   #3,G^LIB$PUT_SCREEN       ;
              30 50  E9  1222        BLBC    R0,DISPLAY_ERROR          ; Branch on error.
        00000205'EF  DF  1225        PUSHAL  FREE_BUFF_MSG             ; Print "CI buffer descriptors free."
     00000000'GF  01 FB  122B        CALLS   #1,G^LIB$PUT_SCREEN       ;
```

```
         20 50    E9  1232               BLBC    R0,DISPLAY_ERROR            ; Branch on error.
 00000000'GF 00   FB  1235               CALLS   #0,G^LIB$ERASE_LINE         ; Erase the rest of the line.
         16 50    E9  123C               BLBC    R0,DISPLAY_ERROR            ; Branch on error.
  00002248'EF     D6  123F               INCL    LINE_NUM                    ; Set line number to next line.
                      1245
                      1245 DISPLAY_END:
                      1245
                      1245 ;
                      1245 ; Erase any lines following the report, and update the scrolling
                      1245 ; signals and broadcast messages at the bottom of the display.
                      1245 ;
                      1245
          0851    30  1245               BSBW    ERASE_PAGE                  ; Erase any remaining data.
         0A 50    E9  1248               BLBC    R0,DISPLAY_ERROR            ; Branch on error.
          0686    30  124B               BSBW    CHECK_MORE                  ; Add scrolling signals or broadcast
                      124E                                                   ;  messages to the report.
 00000476'EF  08  CA  124E               BICL    #MF$M_RIP,MISC_FLAGS        ; Clear refresh in progress flag.
                      1255
                      1255 DISPLAY_ERROR:
       0480 8F    BA  1255               POPR    #^M<R7,R10>
             05       1259               RSB
                      125A
                      125A
```

```
                                    125A  4366 ;
                                    125A  4367 ; This routine is called output of either the Cluster Report or the Local
                                    125A  4368 ; report. It determines if the bottom line of the box surrounding the data
                                    125A  4369 ; should be draw.  This line is drawn only if there is room on the screen and
                                    125A  4370 ; either the line won't be drawn in the same place is was last pass, or the
                                    125A  4371 ; screen is being refreshed.
                                    125A  4372 ;
                                    125A  4373
                                    125A  4374 DISPLAY_COMMON:
                     50    01   D0  125A  4375         MOVL    #SS$_NORMAL,R0           ; Assume success.
  00000428'EF   00002248'EF   D0  125D  4376         MOVL    LINE_NUM,NUM_LINES       ; Save number of lines this pass.
                00000444'EF   D4  1268  4377         CLRL    SCROLL_DIF               ; Clear record of lines to redraw
                                    126E  4378                                          ;  because we scrolled.
                           03   E0  126E  4379         BBS     #SCR$V_MORE_DWN,-        ; Branch if we have more lines of data
             3E 0000047A'EF       1270  4380                 SCROLL_FLAGS,20$         ;  than will fit on the screen.
  0000225C'EF   00002248'EF   D1  1276  4381         CMPL    LINE_NUM,CMD_LINE        ; Will the next line be in the command
                                    1281  4382                                          ;  region?
                           31   13  1281  4383         BEQL    20$                      ; If so don't output the last line.
                           08   E4  1283  4384         BBSC    #MF$V_SCROLL,-           ; Clear flag that indicates that we
             20 00000476'EF       1285  4385                 MISC_FLAGS,10$           ;  have scrolled.
                00000428'EF   D1  128B  4386         CMPL    NUM_LINES,-              ; Do we have the same number of lines
                0000042C'EF       1291  4387                 PRIOR_NUM_LINES          ;  of data as last pass?
                           13   12  1296  4388         BNEQ    10$                      ; If not, draw the last line of the box.
       0B 00000476'EF      03   E0  1298  4389         BBS     #MF$V_RIP,MISC_FLAGS,10$; If refresh in progress better draw the
                                    12A0  4390                                          ;  last line.
                00002248'EF   D6  12A0  4391         INCL    LINE_NUM                 ; Skip the last line draw.
                     50    01   D0  12A6  4392         MOVL    #SS$_NORMAL,R0           ; Indicate success.
                           09   11  12A9  4393         BRB     20$                      ; Jump to common code.
                         0E43   30  12AB  4394 10$:    BSBW    DRAW_LAST_LINE           ; Draw bottom line of display box.
                                    12AE  4395         JLBC    R0,MAIN_ERR              ; Branch on error.
                                05  12B4  4396 20$:    RSB
                                    12B5  4397
```

```
12B5          .SBTTL  SCS report generation routine.
12B5     ;++
12B5     ;
12B5     ; FUNCTIONAL DESCRIPTION:
12B5     ;
12B5     ;       This routine formats the SCS portion of the cluster report.  This
12B5     ;       portion of the report has been designed to illustrate the heirarchy of
12B5     ;       the cluster by showing the relationship between a system on the cluster,
12B5     ;       the circuits established on the system, and the connections established
12B5     ;       over these circuits.  This portion of the report also displays the
12B5     ;       system membership status (MEMBERS class).
12B5     ;
12B5     ;       This routine scans through the snapshot buffer looking for data to be
12B5     ;       displayed while controlling what data is to be displayed on a
12B5     ;       particular line of the report (using the mask at DATA_CNTRL).
12B5     ;       If the data found in the snapshot buffer conforms to the data to be
12B5     ;       displayed on the particular line and the user has elected to see that
12B5     ;       type of data, then a call is made to the GET_DATA routine to extract,
12B5     ;        format and add the line of data to the screen buffer.
12B5     ;
12B5     ;       This routine also sets up the counter V_SCROLL_LINE by using the
12B5     ;       value in V_SCROLL.  This count is an input to the GET_DATA routine
12B5     ;       which causes it to ignore the first "V_SCROLL" times it is called.
12B5     ;       This effectively scrolls the report up "V_SCROLL" lines.
12B5     ;
12B5     ; CALLING SEQUENCE:
12B5     ;
12B5     ;       JSB     SCS_REPORT
12B5     ;
12B5     ; CALLED BY:
12B5     ;
12B5     ;       DISPLAY_SCS
12B5     ;
12B5     ; INPUTS:
12B5     ;
12B5     ;       V_SCROLL        Set to the number of lines of data to skip before
12B5     ;                       adding the first line of data to the report.
12B5     ;
12B5     ;       CLASS_SELECT    Contains a mask of the classes selected for display
12B5     ;                       by the user.
12B5     ;
12B5     ; IMPLICIT INPUTS:
12B5     ;
12B5     ;       Snapshot has been performed and all the system blocks, path blocks, etc.
12B5     ;       have been stored in SNAP_BUF.
12B5     ;
12B5     ; IMPLICIT OUTPUTS:
12B5     ;
12B5     ;       Display data stored in SCREEN_BUF.
12B5     ;
12B5     ; RETURN STATUS:
12B5     ;
12B5     ;       Status from call to GET_DATA routine passed in R0.
12B5     ;
12B5     ; REGISTER USAGE:
12B5     ;
12B5     ;       R8  -   Pointer to connections in CDT buffer.
```

```
                               12B5  ;         R9  -   Pointer to path blocks in PB buffer.
                               12B5  ;         R10 -   Pointer to cluster system block in CSB buffer.
                               12B5  ;         R11 -   Pointer to system blocks in SB buffer.
                               12B5  ;
                               12B5  ;
                               12B5  ; NOTES:
                               12B5  ;         In order to display a class, the corresponding CLS$ (class) bit in
                               12B5  ;         CLASS_SELECT must be set.  The same must be true of DATA_CNTRL or all
                               12B5  ;         the fields in that class will be filled in with blanks.  Normally the
                               12B5  ;         CLASS_SELECT bits are controlled by the user through the ADD class and
                               12B5  ;         REMOVE class commands and the DATA_CNTRL bits are controlled by Show
                               12B5  ;         Cluster to format the display.  When the CLUSTER class is being
                               12B5  ;         displayed, Show Cluster also manipulates the CLASS_SELECT bits.
                               12B5  ;         CLASS_SELECT bits are manipulated in this routine and in the REFRESH
                               12B5  ;         routine to control the display of the header.
                               12B5  ;--
                               12B5
                               12B5  ASSUME  CLS$V_750_SYS EQ 16
                               12B5
                               12B5  SCS_REPORT:
              0F04 8F      BB  12B5          PUSHR   #^M<R2,R8,R9,R10,R11>
                           CA  12B9          BICL    #<CLS$M_CSB!CLS$M_CIR!-  ; Assume no members, circuits,
                               12BA                    CLS$M_CON!CLS$M_CNT!-  ;  connections, counters, or credits,
                               12BA                    CLS$M_CRD!CLS$M_CLUB>,-; and no general cluster information
000021F8'EF    0000007D 8F     12BA                  DATA_CNTRL               ;  to display, so disable them all.
         5B    000003ED'EF D0  12C4          MOVL    VAR_BUF,R11              ; Get address of first system block.
         52    000004B2'EF DE  12CB  10$:    MOVAL   HW_TYPE_TABLE,R2         ; Point to table of system types.
                       51  D4  12D2          CLRL    R1                       ; Initialize index into table.
            34 AB    6241  D1  12D4  20$:    CMPL    (R2)[R1],-               ; Is the type of system specified by
                               12D9                  SB$T_HWTYPE(R11)         ;  this system block in the table?
                       06  13  12D9          BEQL    30$                      ; Branch if so.
              F5 51    06  F2  12DB          AOBLSS  #NUM_HW_TYPES,R1,20$     ; Loop for all types of systems.
                               12DF  ;
                               12DF  ; If not in the table, accept it anyway.  It could be the local system block
                               12DF  ; and VMSVAXCLUSTER <> 1, or pre VMS Version 3.3 system.
                               12DF  ;
                       0E  11  12DF          BRB     40$                      ;
                               12E1
                 51    10  C0  12E1  30$:    ADDL    #CLS$V_750_SYS,R1        ; Convert index to CLS$ bit value.
                               12E4          JBC     R1,CLASS_SELECT,160$     ; Branch if user has removed this
                               12EF                                           ;  type of system from the display.
        000014F8'EF        16  12EF  40$:    JSB     CHECK_ID                 ; Check if this systems ID is in the
                               12F5                                           ;  disabled list.
                               12F5          JLBS    R0,160$                  ; Go on to next system block if it is.
     000021F8'EF   02      C8  12FB          BISL    #CLS$M_SYS,DATA_CNTRL    ; Enable display of systems class.
        00002248'EF        90  1302          MOVB    LINE_NUM,-               ; Store line number for this system in
                   09 AB       1308                  SHC$SB$B_DIS_POS(R11)    ;  system block.
                       02  E1  130A  50$:    BBC     #CLS$V_CSB,-             ; Branch if user has removed members
      14 00000001'EF           130C                  CLASS_SELECT,55$         ;  from the display.
     000021F8'EF   04      CA  1312          BICL    #CLS$M_CSB,DATA_CNTRL    ; Assume no CSB associated with this
                               1319                                           ;  system.
            5A    5C AB    D0  1319          MOVL    SB$L_CSB(R11),R10        ; Get address of CSB for this system.
                       07  13  131D          BEQL    55$                      ; Branch if no CSB.
     000021F8'EF   04      C8  131F          BISL    #CLS$M_CSB,DATA_CNTRL    ; Enable display of CSB's.
        0000144C'EF        16  1326  55$:    JSB     CHECK_NUM_PB             ; See if the number of path blocks on
                               132C                                           ;  this system block has changed.
            59    0C AB    D0  132C          MOVL    SB$L_PBFL(R11),R9        ; Get address of first path block on
```

```
                              1330                                          ;  this system block.
                    6B   13   1330               BEQL    120$               ; Branch if there are no circuits on
                              1332                                          ;  this system.
                 12 A9   B1   1332               CMPW    PB$W_STATE(R9),-   ; Is the circuit in the open state?
                    03        1335                       #PB$C_OPEN         ;
                    0A   12   1336               BNEQ    70$                ; Branch if not.
                    09   E1   1338       60$:    BBC     #CLS$V_OPEN_CIR,-  ; Branch if user has removed open
        11 00000001'EF        133A                       CLASS_SELECT,90$   ;  circuits from the display.
                    08   11   1340               BRB     80$                ;
                    0A   E1   1342       70$:    BBC     #CLS$V_NOOPEN_CIR,- ; Branch if user has removed circuits in
        07 00000001'EF        1344                       CLASS_SELECT,90$   ;  states other that open from the
                              134A                                          ;  display.
   000021F8'EF      08   C8   134A       80$:    BISL    #CLS$M_CIR,DATA_CNTRL ; Enable display of circuits class.
      0000147A'EF        16   1351       90$:    JSB     CHECK_NUM_CDT      ; See if the number of connections
                              1357                                          ;  on this path has changed since the
                              1357                                          ;  last pass.
           58    34 A9   D0   1357               MOVL    PB$L_CDTLST(R9),R8 ; Get address of first CDT on this
                              135B                                          ;  path block.
                    40   13   135B               BEQL    120$               ; Branch if there are no connections
                              135D                                          ;  on this circuit.
                 18 A8   B1   135D               CMPW    SHC$L_CDT_DIF+CDT$W_STATE(R8),-
                    01        1360                       #CDT$C_LISTEN      ; Is this CDT in the ''listen'' state?
                    10   12   1361               BNEQ    110$               ; Branch if not.
   000021F8'EF      08   CA   1363               BICL    #CLS$M_CIR,DATA_CNTRL ; Otherwise, disable display of
                              136A                                          ;  circuits class.
   000021F8'EF      10   C8   136A               BISL    #CLS$M_CON,DATA_CNTRL ; Enable display of connections class.
                    2A   11   1371               BRB     120$               ; Jump to common code.
                 18 A8   B1   1373       110$:   CMPW    SHC$L_CDT_DIF+CDT$W_STATE(R8),-
                    02        1376                       #CDT$C_OPEN        ; Is this CDT in the ''open'' state?
                    0A   12   1377               BNEQ    115$               ; Branch if not.
                    0B   E1   1379               BBC     #CLS$V_OPEN_CON,-  ; Branch if user has removed open
        11 00000001'EF        137B                       CLASS_SELECT,119$  ;  open connections from the display.
                    08   11   1381               BRB     118$               ; Branch to common.
                    0C   E1   1383       115$:   BBC     #CLS$V_NOOPEN_CON,- ; Branch if user has removed non-open
        07 00000001'EF        1385                       CLASS_SELECT,119$  ;  connections from the display.
   000021F8'EF      10   C8   138B       118$:   BISL    #CLS$M_CON,DATA_CNTRL ; Enable display of connections class.
      00000060      8F   C8   1392       119$:   BISL    #CLS$M_CRD!CLS$M_CNT,- ; Enable display of counters and
   000021F8'EF                1398                       DATA_CNTRL         ;  credits classes.
                              139D
                              139D       ;
                              139D       ; AND the mask of classes we've enabled with the mask of classes the user has
                              139D       ; selected and see if there's any data to be displayed.  If so, output a line
                              139D       ; of data, otherwise continue looking.
                              139D       ;
                              139D
   51   000021F8'EF      D2   139D       120$:   MCOML   DATA_CNTRL,R1      ; Get mask of classes enabled.
51    00000001'EF   51   CB   13A4               BICL3   R1,CLASS_SELECT,R1 ; Any data to be displayed?
                    2A   13   13AC               BEQL    130$               ; Branch if not.
   00000476'EF      03   E0   13AE               BBS     #MF$V_RIP,MISC_FLAGS,- ; Skip position check if performing a
                    0E        13B5                       125$               ;  refresh.
                    08   E0   13B6               BBS     #MF$V_SCROLL,-     ; Skip position check if we just
        06 00000476'EF        13B8                       MISC_FLAGS,125$    ;  scrolled.
      000014A8'EF        16   13BE               JSB     CHECK_POS          ; Check if the system has moved in the
                              13C4                                          ;  display.
                    58   DD   13C4       125$:   PUSHL   R8                 ; Push address of CDT.
                    59   DD   13C6               PUSHL   R9                 ; Push address of path block.
                    5A   DD   13C8               PUSHL   R10                ; Push address of cluster system block.
```

```
                          5B  DD  13CA         PUSHL   R11                          ; Push address of system block.
                          00  DD  13CC         PUSHL   #0                           ; Bogus CLUB address.
            00002729'EF   05  FB  13CE         CALLS   #5,GET_DATA                  ; Get a line of data.
                       6F 50  E9  13D5         BLBC    R0,999$                      ; Branch on error.
 000021F8'EF   00000070 8F    D3  13D8  130$:  BITL    #<CLS$M_CON!CLS$M_CNT!-      ; Are we displaying connections,
                                  13E3                  CLS$M_CRD>,DATA_CNTRL       ;  counters or credits?
                          0C  13  13E3         BEQLU   140$                         ; Branch if not.
            000021F8'EF   0E  CA  13E5         BICL    #<CLS$M_SYS!CLS$M_CSB!-      ; Disable display of systems, members,
                                  13EC                  CLS$M_CIR>,DATA_CNTRL       ;  and circuits class.
                       5B 68  D0  13EC         MOVL    (R8),R8                      ; Get address of next CDT.
                          AC  12  13EF         BNEQ    120$                         ; Loop for all connections on this
                                  13F1                                              ;  circuit.
 000021F8'EF   00000070 8F    CA  13F1  140$:  BICL    #<CLS$M_CON!CLS$M_CNT!-      ; Assume no connections on the next
                                  13FC                  CLS$M_CRD>,DATA_CNTRL       ;  circuit, so disable display of
                                  13FC                                              ;  connections, counters, and credits
                                  13FC                                              ;  classes.
                          59  D5  13FC         TSTL    R9                           ; Are there any circuits on this system?
                          1A  13  13FE         BEQL    160$                         ; Branch if not.
                       59 69  D0  1400         MOVL    (R9),R9                      ; Get address of next path block.
                          0A  13  1403         BEQL    150$                         ; Branch if no more circuits on this
                                  1405                                              ;  system.
                          06  CA  1405         BICL    #CLS$M_SYS!CLS$M_CSB,-       ; Disable display of systems, members
               000021F8'EF        1407          DATA_CNTRL                          ;  classes.
                        FF29  31  140C         BRW     60$                          ; Loop for all circuits on this system.
                              CA  140F  150$:  BICL    #<CLS$M_CIR!CLS$M_CON!-      ; Assume no circuits on next system, so
                                  1410                  CLS$M_CNT!CLS$M_CRD>,-      ;  disable display of circuits,
 000021F8'EF   00000078 8F        1410          DATA_CNTRL                          ;  connections, counters, and credits.
          51   000001F3'EF    DE  141A  160$:  MOVAL   DUMMY_SB,R1                  ; Determine if we are displaying
                       5B 51  D1  1421         CMPL    R1,R11                       ;  formative paths without formative
                                  1424                                              ;  system blocks.
                          21  13  1424         BEQL    999$                         ; If so, end the report.
                       5B 6B  D0  1426         MOVL    (R11),R11                    ; Get address of next system block.
                                  1429         JNEQ    10$                          ; Loop for all systems in the cluster.
                                  142E  ;
                                  142E  ; Display any formative path blocks that have no associated formative system
                                  142E  ; block.
                                  142E  ;
          5B   000001F3'EF    DE  142E         MOVAL   DUMMY_SB,R11                 ; Get the address of the fake system
                                  1435                                              ;  block.
                       0C AB  D5  1435         TSTL    SB$L_PBFL(R11)               ; Are there any formative path blocks
                                  1438                                              ;  on the dummy system block?
                          0D  13  1438         BEQL    999$                         ; Branch if not.
                        10E3  30  143A         BSBW    GET_BLANK                    ; Leave one blank line.
                          06  CA  143D         BICL    #CLS$M_SYS!CLS$M_CSB,-       ; Disable display of systems, and
               000021F8'EF        143F          DATA_CNTRL                          ;  members classes of data.
                        FEC3  31  1444         BRW     50$                          ; Display the formative path blocks.
                                  1447
                     0F04 8F  BA  1447  999$:  POPR    #^M<R2,R8,R9,R10,R11>
                          05      144B         RSB
                                  144C
                                  144C
```

```
                                144C  ;++
                                144C  ;
                                144C  ; FUNCTIONAL DESCRIPTION:
                                144C  ;
                                144C  ;       This routine insures that a change in the number of path blocks
                                144C  ;       does not go undetected.  When a path block is removed from the
                                144C  ;       database, CHECK_DIF is no longer called to check for a change
                                144C  ;       in data for that path block.  This means that the line will
                                144C  ;       not be redrawn until the screen is refreshed.  This routine
                                144C  ;       calls CHECK_DIF with the data to be checked being the number
                                144C  ;       of path blocks, so a change will always be detected.
                                144C  ;
                                144C  ; CALLING SEQUENCE:
                                144C  ;
                                144C  ;       JSB     CHECK_NUM_PB
                                144C  ;
                                144C  ; CALLED BY:
                                144C  ;
                                144C  ;       CLUSTER_REPORT
                                144C  ;
                                144C  ; INPUTS:
                                144C  ;
                                144C  ;       Address of the system block in R11
                                144C  ;
                                144C  ; OUTPUTS:
                                144C  ;
                                144C  ;       The bit coresponding to the current line number in LINE_DIF
                                144C  ;       is set if the number of path block on this system block has
                                144C  ;       changed.
                                144C  ;
                                144C  ;--
                                144C
                                144C  CHECK_NUM_PB:
              0480 8F      BB   144C          PUSHR   #^M<R7,R10>
            57   04 AB     DE   1450          MOVAL   SB$SHC$L_NUM_PB(R11),R7 ; Get the address of the data.
            000001A9'EF    DE   1454          MOVAL   SB$SHC$L_NUM_PB_DSCR,-  ; Get the address of the field
                    5A          145A                  R10                     ;  descriptor.
            00002D3D'EF    16   145B          JSB     CHECK_DIF               ; See if the number of path
                                1461                                          ;  blocks changed.
                    02     E5   1461          BBCC    #MF$V_INVERSE,-         ; Branch if there was no change.
         0C 00000476'EF         1463                  MISC_FLAGS,99$          ;
00 00000434'EF 00002248'EF E2   1469          BBSS    LINE_NUM,LINE_DIF,99$   ; Indicate that this line of data
                                1475                                          ;  must be redrawn.
              0480 8F      BA   1475  99$:    POPR    #^M<R7,R10>
                    05          1479          RSB
                                147A
```

```
                                     147A  ;++
                                     147A  ;
                                     147A  ; FUNCTIONAL DESCRIPTION:
                                     147A  ;
                                     147A  ;      This routine insures that a change in the number of connections
                                     147A  ;      does not go undetected.  When a CDT is removed from the database
                                     147A  ;      CHECK_DIF is no longer called to check for a change in data for
                                     147A  ;      that CDT.  This means that the line will not be redrawn until
                                     147A  ;      screen is refreshed.  This routine calls CHECK_DIF with the data
                                     147A  ;      to be checked being the number of CDTS , so a change will always
                                     147A  ;      be detected.
                                     147A  ;
                                     147A  ; CALLING SEQUENCE:
                                     147A  ;
                                     147A  ;      JSB      CHECK_NUM_CDT
                                     147A  ;
                                     147A  ; CALLED BY:
                                     147A  ;
                                     147A  ;      CLUSTER_REPORT
                                     147A  ;
                                     147A  ; INPUTS:
                                     147A  ;
                                     147A  ;      Address of Path Block in R11
                                     147A  ;
                                     147A  ; OUTPUTS:
                                     147A  ;
                                     147A  ;      The bit coresponding to the current line number in LINE_DIF
                                     147A  ;      is set if the number of CDT's on this path block has changed.
                                     147A  ;
                                     147A  ;--
                                     147A
                                     147A  CHECK_NUM_CDT:
                    0480 8F     BB   147A         PUSHR   #^M<R7,R10>
                 57   04 AB     DE   147E         MOVAL   PB$SHC$L_NUM_CDT(R11),R7; Get the address of the data.
               00000501'EF      DE   1482         MOVAL   PB$SHC$L_NUM_CDT_DSCR,- ; Get the address of the field
                         5A          1488                 R10                     ;  descriptor.
               00002D3D'EF      16   1489         JSB     CHECK_DIF               ; See if the number of path
                                     148F                                         ;  blocks changed.
                         02     E5   148F         BBCC    #MF$V_INVERSE,-         ; Branch if there was no change.
            0C 00000476'EF           1491                 MISC_FLAGS,99$          ;
00 00000434'EF 00002248'EF      E2   1497         BBSS    LINE_NUM,LINE_DIF,99$   ; Indicate that this line of data
                                     14A3                                         ;  must be redrawn.
                    0480 8F     BA   14A3  99$:   POPR    #^M<R7,R10>
                         05          14A7         RSB
                                     14A8
```

```
                              14A8  ;++
                              14A8  ;
                              14A8  ; FUNCTIONAL DESCRIPTION:
                              14A8  ;
                              14A8  ;       This routine compares the display line number stored in the system block
                              14A8  ;       at SHC$SB$B_DIS_POS  (SB$W_SIZE+1)  with the display line number stored
                              14A8  ;       during the last update.  If different it indicates that the data for
                              14A8  ;       this system has moved to a different part of the display.  If the data
                              14A8  ;       itself has has not changed, then this could otherwise go undetected.
                              14A8  ;
                              14A8  ; CALLING SEQUENCE:
                              14A8  ;
                              14A8  ;       JSB     CHECK_POS
                              14A8  ;
                              14A8  ; CALLED BY:
                              14A8  ;
                              14A8  ;       CLUSTER_REPORT
                              14A8  ;
                              14A8  ; INPUTS:
                              14A8  ;
                              14A8  ;       R11 = Base address of system block containing line number to check.
                              14A8  ;       OLD_VAR_BUF points to buffer containing system blocks from the previous
                              14A8  ;       pass.
                              14A8  ;
                              14A8  ; OUTPUTS:
                              14A8  ;
                              14A8  ;       If the line numbers are found to be different, then a bit corresponding
                              14A8  ;       to the line number is set in LINE_DIF.
                              14A8  ;
                              14A8  ; RETURN STATUS:
                              14A8  ;
                              14A8  ;       R0 = SS$_NORMAL
                              14A8  ;
                              14A8  ;--
                              14A8
                              14A8  CHECK_POS:
                   0C    BB   14A8          PUSHR   #^M<R2,R3>
      000001F3'8F  5B    D1   14AA          CMPL    R11,#DUMMY_SB           ; Does R11 point to the fake system
                              14B1                                          ;  block?
                   14    12   14B1          BNEQ    10$                     ; Branch if not.
    53  000004A3'EF      DE   14B3          MOVAL   OLD_FIX_BUF,R3          ; Point R3 to previous fixed length
                              14BA                                          ;  buffers.
    52  0000004B'EF      DE   14BA          MOVAL   FIX_BUF,R2              ; Get the address of the current fixed
                              14C1                                          ;  length buffers.
          52    5B   52  C3   14C1          SUBL3   R2,R11,R2               ; Get the offset into the current
                              14C5                                          ;  fixed length buffer where this
                              14C5                                          ;  system block is located.
                   18    11   14C5          BRB     20$                     ; Branch to common.
    53  00000419'EF      D0   14C7  10$:    MOVL    OLD_VAR_BUF,R3          ; Point R3 to the previous variable
                              14CE                                          ;  length buffers.
52  5B  000003ED'EF      C3   14CE          SUBL3   VAR_BUF,R11,R2          ; Get the offset into the current
                              14D6                                          ;  variable length buffer where this
                              14D6                                          ;  system block is located.
      00000421'EF  52    D1   14D6          CMPL    R2,OLD_VAR_BUF_LEN      ; Is the offset larger than length of
                              14DD                                          ;  the previous buffer?
                   0A    18   14DD          BGEQ    30$                     ; Branch if so.
             53    52    C0   14DF  20$:    ADDL    R2,R3                   ; Locate the system block in the
```

```
                                      14E2  4772                                              ;  old buffer.
                        09 AB    91   14E2  4773              CMPB    SHC$SB$B_DIS_POS(R11),-  ; Is the line number different?
                        09 A3         14E5  4774                      SHC$SB$B_DIS_POS(R3)     ;
                           0C    13   14E7  4775              BEQL    40$                      ; Branch if not.
 00 00000434'EF    00002248'EF   E2   14E9  4776  30$:        BBSS    LINE_NUM,LINE_DIF,40$    ; Indicate line must be redrawn.
                           0C    BA   14F5  4777  40$:        POPR    #^M<R2,R3>
                                 05   14F7  4778              RSB
                                      14F8  4779
                                      14F8  4780
```

```
                  14F8  ;++
                  14F8  ;
                  14F8  ; FUNCTIONAL DESCRIPTION:
                  14F8  ;
                  14F8  ;       This routine takes the system ID found in the system block and calls
                  14F8  ;       FIND_SYS_ID to check if the system ID is in the SYS_ID_LIST of system
                  14F8  ;       that are not to be display.  It returns success if the system ID was
                  14F8  ;       found.
                  14F8  ;
                  14F8  ; CALLING SEQUENCE:
                  14F8  ;
                  14F8  ;       JSB     CHECK_ID
                  14F8  ;
                  14F8  ; CALLED BY:
                  14F8  ;
                  14F8  ;       CLUSTER_REPORT
                  14F8  ;
                  14F8  ; INPUTS:
                  14F8  ;
                  14F8  ;       R11 = Base address of system block containing ID to check.
                  14F8  ;
                  14F8  ;
                  14F8  ; RETURN STATUS:
                  14F8  ;
                  14F8  ;       R0 = SS$_NORMAL if system ID is in the list.
                  14F8  ;
                  14F8  ;--
                  14F8
                  14F8  CHECK_ID:
         10   BB  14F8          PUSHR   #^M<R4>                 ; R4 destroyed by FIND_SYS_ID.
      18 AB   D0  14FA          MOVL    SB$B_SYSTEMID(R11),-    ; Get the system ID from the system
  000022DC'EF     14FD                  ID_RESULT               ;  block and store in ID_RESULT.
      1C AB   B0  1502          MOVW    SB$B_SYSTEMID+4(R11),-  ;
  000022E0'EF     1505                  ID_RESULT+4             ;
  00000E16'EF 16  150A          JSB     FIND_SYS_ID             ; Look for it in the list.
         10   BA  1510          POPR    #^M<R4>                 ; Restore R4.
              05  1512          RSB                             ; Return with status from FIND_SYS_ID.
                  1513
                  1513
```

```
                             1513        .SBTTL  Local report generation routine.
                             1513 ;++
                             1513 ;
                             1513 ; FUNCTIONAL DESCRIPTION:
                             1513 ;
                             1513 ;       This routine formats the display for the local port report.  It simply
                             1513 ;       enables both classes in the local report gets the address of the
                             1513 ;       information for the port PDT_BUF and MISC_BUF and calls the GET_DATA
                             1513 ;       routine.  It then determines if a second report is present and if so
                             1513 ;       repeats the process for the second report.
                             1513 ;
                             1513 ;       Note that V_SCROLL_LINE is cleared for this report since it is never
                             1513 ;       necessary to scroll this report vertically.
                             1513 ;
                             1513 ; CALLING SEQUENCE:
                             1513 ;
                             1513 ;       JSB     LOCAL_REPORT
                             1513 ;
                             1513 ; CALLED BY:
                             1513 ;
                             1513 ;       MAIN_LOOP
                             1513 ;
                             1513 ; INPUTS:
                             1513 ;
                             1513 ;       CLASS_SELECT    Contains a mask of the classes selected for display
                             1513 ;                       by the user.
                             1513 ;
                             1513 ; IMPLICIT INPUTS:
                             1513 ;
                             1513 ;       Snapshot has been performed and all the port discriptors and
                             1513 ;       miscellaneous port information has been stored in SNAP_BUF.
                             1513 ;
                             1513 ; IMPLICIT OUTPUTS:
                             1513 ;
                             1513 ;       Display data stored in SCREEN_BUF.
                             1513 ;
                             1513 ; RETURN STATUS:
                             1513 ;
                             1513 ;       Status from call to GET_DATA routine passed in R0.
                             1513 ;
                             1513 ; REGISTER USAGE:
                             1513 ;
                             1513 ;       R9  -   Pointer into MISC_BUF.
                             1513 ;       R10 -   Pointer into PDT_BUF.
                             1513 ;
                             1513 ;--
                             1513
                             1513 LOCAL_REPORT:
           0600 8F     BB    1513        PUSHR   #^M<R9,R10>
5A    000002A7'EF     DE    1517        MOVAL   PDT_BUF,R10             ; Get address of first PDT.
59    00000447'EF     DE    151E        MOVAL   MISC_BUF,R9             ; Get address of miscellaneous buffer.
      00000180 8F     D0    1525        MOVL    #CLS$M_LPRT!CLS$M_ERR,- ; Enable display of local ports and
      000021F8'EF           152B                DATA_CNTRL              ;  errors.
                69     D5    1530        TSTL    (R9)                    ; Any data on first port?
                05     13    1532        BEQL    10$                     ; Branch if not.
                14     10    1534        BSB     100$                    ; Get data for this port.
             0C 50     E9    1536        BLBC    R0,99$                  ; Branch on error.
```

```
              59  2E   C0 1539  4879 10$:   ADDL    #MISC_BUF_LEN,R9        ; Get address of miscellaneous data for
                          153C  4880                                        ;  the second port.
     5A  000000D0  8F   C0 153C  4881        ADDL    #SHC$L_PDT_LENGTH,R10   ; Get address of PDT for second port.
                  05   10 1543  4882        BSB     100$                    ; Get data for this port.
            0600  8F   BA 1545  4883 99$:   POPR    #^M<R9,R10>
                       05 1549  4884        RSB
                          154A  4885
                  69   D5 154A  4886 100$:  TSTL    (R9)                    ; Any data on port?
                  0E   13 154C  4887        BEQL    199$                    ; If not return.
                  59   DD 154E  4888        PUSHL   R9                      ; Save address of miscellaneous buffer.
                  5A   DD 1550  4889        PUSHL   R10                     ; Save address of PDT
              5E  14   C2 1552  4890        SUBL    #5*4,SP                 ; Adjust SP for no CDT, PB, CSB, SB, or
                          1555  4891                                        ;  CLUB.
     00002729'EF  07   FB 1555  4892        CALLS   #7,GET_DATA             ; Get info on port.
                       05 155C  4893 199$:  RSB
                          155D  4894
                          155D  4895
```

```
                                   155D        .SBTTL  Error message output.
                                   155D  ;++
                                   155D  ;
                                   155D  ; FUNCTIONAL DESCRIPTION:
                                   155D  ;
                                   155D  ;       This routine does a call to $GETMSG using the status stored in
                                   155D  ;       CMD_STATUS. (normally the result from the last command parse).
                                   155D  ;       It then determines if any FAO arguments are expected with the
                                   155D  ;       message and sets up the message vector accordingly.  Finally it
                                   155D  ;       does a call to LIB$SIGNAL to output the message to the terminal.
                                   155D  ;
                                   155D  ; CALLING SEQUENCE:
                                   155D  ;
                                   155D  ;       JSB     SIGNAL
                                   155D  ;
                                   155D  ; CALLED BY:
                                   155D  ;
                                   155D  ;       READ_W_PROMPT
                                   155D  ;       SCROLL_TWO
                                   155D  ;
                                   155D  ; INPUTS:
                                   155D  ;
                                   155D  ;       Message identification number stored at CMD_STATUS.
                                   155D  ;
                                   155D  ; IMPLICIT IMPUTS:
                                   155D  ;
                                   155D  ;       Descriptor for FAO argument stored at TPARSE_BLK+TPA$L_STRINGCNT.
                                   155D  ;
                                   155D  ; OUTPUTS:
                                   155D  ;
                                   155D  ;       Message output to the terminal.
                                   155D  ;
                                   155D  ; RETURN STATUS:
                                   155D  ;
                                   155D  ;       SS$_NORMAL or failed status returned in R0.
                                   155D  ;
                                   155D  ;--
                                   155D
                                   155D  SIGNAL:
              000004B9'EF    D1    155D         CMPL    CMD_STATUS,-            ; Syntax error detected by TPARSE?
              00000000'8F          1563                 #LIB$_SYNTAXERR         ;
                       0D    12    1568         BNEQ    10$                     ; Branch if not.
000004B9'EF   00000000'8F    D0    156A         MOVL    #SHOW$_UNRCMD,CMD_STATUS; "Unrecognized command"
                       32    11    1575         BRB     30$                     ; Branch to common code.
                                   1577
00000000'8F   000004B9'EF    D1    1577  10$:   CMPL    CMD_STATUS,#LIB$_UNRKEY ; Unrecognized keyword?
                       0D    12    1582         BNEQ    20$                     ; Branch if not.
000004B9'EF   00000000'8F    D0    1584         MOVL    #SHOW$_UNRKEY,CMD_STATUS; "Unrecognized keyword"
                       18    11    158F         BRB     30$                     ; Branch to common code.
                                   1591
00000000'8F   000004B9'EF    D1    1591  20$:   CMPL    CMD_STATUS,#LIB$_AMBKEY ; Ambiguous keyword?
                       0B    12    159C         BNEQ    30$                     ; Branch if not.
000004B9'EF   00000000'8F    D0    159E         MOVL    #SHOW$_AMBIG,CMD_STATUS ; "Ambiguous keyword"
                                   15A9
                                   15A9  30$:   $GETMSG_S -                     ; Get the error message into the
                                   15A9                 MSGID = CMD_STATUS,-    ;  buffer.
                                   15A9                 MSGLEN = ERR_MSG_DSCR,- ;
```

```
                              15A9  4954                     BUFADR = GET_MSG_DSCR,-  ;
                              15A9  4955                     OUTADR = MSG_OUT_ADRS    ;
             4C 50      E9    15CA  4956          BLBC       R0,99$                   ; Branch on error.
                              15CD  4957                                              ;
      000004AE'EF       95    15CD  4958          TSTB       MSG_OUT_ADRS+1           ; Is there an FAO argument associated
                              15D3  4959                                              ;  with the message.
               2E       13    15D3  4960          BEQL       40$                      ; Branch if not.
  0000049D'EF   03      B0    15D5  4961          MOVW       #3,ERR_MSGVEC            ; Otherwise set the number of longwords
                              15DC  4962                                              ;  in the message vector to include the
                              15DC  4963                                              ;  FAO argument.
      000004AE'EF       90    15DC  4964          MOVB       MSG_OUT_ADRS+1,-         ; Save the number of FAO arguments in
      000004A5'EF             15E2  4965                     ERR_MSGVEC+8             ;  the message vector.
      000004A9'EF       DD    15E7  4966          PUSHL      ERR_MSGVEC+12            ;
  7E  000004A5'EF       3C    15ED  4967          MOVZWL     ERR_MSGVEC+8,-(SP)       ; Output the error message with FAO
      000004B9'EF       DD    15F4  4968          PUSHL      CMD_STATUS               ;  directives.
  00000000'GF   03      FB    15FA  4969          CALLS      #3,G^LIB$SIGNAL          ;
               16       11    1601  4970          BRB        99$                      ; Branch to common code.
                              1603  4971                                              ;
  0000049D'EF   01      90    1603  4972 40$:     MOVB       #1,ERR_MSGVEC            ; Set the number of longwords in the
                              160A  4973                                              ;  message vector to indicate that
                              160A  4974                                              ;  there are no FAO arguments are
                              160A  4975                                              ;  associated with the message.
      000004B9'EF       DD    160A  4976          PUSHL      CMD_STATUS               ; Output the error message.
  00000000'GF   01      FB    1610  4977          CALLS      #1,G^LIB$SIGNAL          ;
               00       11    1617  4978          BRB        99$                      ; Branch to common code.
                              1619  4979
                        05    1619  4980 99$:     RSB
                              161A  4981
                              161A  4982
```

```
                              161A        .SBTTL  Broadcast message handling.
                              161A  ;++
                              161A  ;
                              161A  ; FUNCTIONAL DESCRIPTION:
                              161A  ;
                              161A  ;       This routine is called to output a broadcast message.  It determines
                              161A  ;       if any information is being display on the bottom line of the report
                              161A  ;       (command input, error messages, or a previous broadcast message).
                              161A  ;       If no information is being displayed, then a line of a broadcast
                              161A  ;       message is removed from the MSG_OUT_QUE (if available) and displayed
                              161A  ;       on the bottom line.  Once output, the buffer used for the message is
                              161A  ;       placed in the MSG_FRE_QUE to save calls to GET_VM by the STORE_MSG
                              161A  ;       routine.
                              161A  ;
                              161A  ; CALLING SEQUENCE:
                              161A  ;
                              161A  ;       JSB     OUT_MSG
                              161A  ;
                              161A  ; CALLED BY:
                              161A  ;
                              161A  ;       CHECK_MORE
                              161A  ;
                              161A  ; IMPLICIT INPUTS:
                              161A  ;
                              161A  ;       BOT_CMD_LINE    Indicates the type of information currently being
                              161A  ;                       displayed in the bottom line of the command scroll
                              161A  ;                       region.
                              161A  ;
                              161A  ;       MSG_OUT_QUE     Contains a list of buffers containing message lines
                              161A  ;                       to be output.
                              161A  ;
                              161A  ; IMPLICIT OUTPUTS:
                              161A  ;
                              161A  ;       Broadcast message output to the screen buffer.
                              161A  ;       Message buffer added to MSG_FRE_QUE
                              161A  ;       Message buffer removed from MSG_OUT_QUE
                              161A  ;
                              161A  ;       BOT_CMD_LINE    Set to indicate that there is a broadcast message on
                              161A  ;                       the bottom line of the command scroll region if a
                              161A  ;                       message was output.
                              161A  ;
                              161A  ; RETURN STATUS:
                              161A  ;
                              161A  ;       SS$_NORMAL or failed status returned in R0.
                              161A  ;
                              161A  ;--
                              161A
                    3C   BB   161A  OUT_MSG:PUSHR   #^M<R2,R3,R4,R5>
           00000471'EF   95   161C          TSTB    BOT_CMD_LINE            ; Any information on the bottom command
                              1622                                          ;  line?
                              1622          JNEQ    99$                     ; Branch if so.
00000000'EF  00000010'FF  0F  1627          REMQUE  @MSG_OUT_QUE,BUFADR     ; Remove a line from the message output
                              1632                                          ;  queue.
                    03   1C   1632          BVC     10$                     ; Skip next if available.
                  007F   31   1634          BRW     99$                     ; Branch if none available.
    54  00000000'EF  08  C1   1637  10$:    ADDL3   #8,BUFADR,R4            ; Get the address of the message.
                              163F                                          ; Build message descriptor
```

```
          00001FAD'EF   84  3C  163F       MOVZWL  (R4)+,MBX_MSG_DSCR           ;  - length and type
          00001FB1'EF   54  D0  1646       MOVL    R4,MBX_MSG_DSCR+4            ;  - address.
00002248'EF  00002260'EF    D0  164D       MOVL    LAST_LINE,LINE_NUM           ; Output message on the bottom line.
             0000224C'EF    3F  1658       PUSHAW  COL_NUM                      ; Output the broadcast message.
             00002248'EF    3F  165E       PUSHAW  LINE_NUM                     ;
             00001FAD'EF    3F  1664       PUSHAW  MBX_MSG_DSCR                 ;
          00000000'GF   03  FB  166A       CALLS   #3,G^LIB$PUT_SCREEN          ;
                     42 50  E9  1671       BLBC    R0,99$                       ; Branch on error.
          00000000'GF   00  FB  1674       CALLS   #0,G^LIB$ERASE_LINE          ; Erase the rest of the line.
                     38 50  E9  167B       BLBC    R0,99$                       ; Branch on error.
       1B 00002247'EF   01  E0  167E       BBS     #DVT$V_DCRT,OUT_DEV,20$      ; Branch if DECcrt terminal.
             00001FAD'EF    28  1686       MOVC3   MBX_MSG_DSCR,-               ; Save a copy of the last message
             00001FB1'EF        168C               MBX_MSG_DSCR+4,-             ;  output so that in can be scrolled.
             00001DFA'EF        1691               LAST_BRDCST_BUF              ;
             00001FAD'EF    D0  1696       MOVL    MBX_MSG_DSCR,-               ; Save the length of the message.
             00001DF2'EF        169C               LAST_BRDCST_DSCR             ;
                        01  88  16A1 20$:  BISB    #CLS$M_BRDCST,-              ; Indicate that there is a broadcast
             00000471'EF        16A3               BOT_CMD_LINE                 ;  message on the bottom command line.
                     50 01  D0  16A8       MOVL    #SS$_NORMAL,R0
00000008'EF  00000000'FF    0E  16AB       INSQUE  @BUFADR,MSG_FRE_QUE          ; Place message buffer in message free
                                16B6                                            ;  queue.
                        3C  BA  16B6 99$:  POPR    #^M<R2,R3,R4,R5>
                            05  16B8       RSB
                                16B9
                                16B9
```

```
                                 16B9           .SBTTL  Scrolling routines.
                                 16B9           .SBTTL          -       SET_SCROLL routine.
                                 16B9
                                 16B9           .ENABL  LSB
                                 16B9
                                 16B9   ;++
                                 16B9   ;
                                 16B9   ; FUNCTIONAL DESCRIPTION:
                                 16B9   ;
                                 16B9   ;       This routine translates the characters following an escape in the
                                 16B9   ;       command buffer into a scroll command.  If the escape sequence indicates
                                 16B9   ;       that one of the arrow keys has been pressed then the value stored at
                                 16B9   ;       location V_SCROLL (for vertical scrolling) or either CLSTR_H_SCROLL
                                 16B9   ;       or LOCAL_H_SCROLL (for horizontal scrolling depending on which report
                                 16B9   ;       is being displayed) is incremented or decremented appropriately.
                                 16B9   ;
                                 16B9   ;       This routine also sets flags at location SCROLL_FLAGS if it determines
                                 16B9   ;       that data has been scrolled off the left or top of the screen or clears
                                 16B9   ;       the flags if the data has been restored.
                                 16B9   ;
                                 16B9   ;       If the terminal type is DECcrt, and the display is scrolled up or
                                 16B9   ;       down, then bits are set in SCROLL_DIF corresponding to the lines
                                 16B9   ;       which much be redrawn to complete the display.
                                 16B9   ;
                                 16B9   ;       NOTE:   This routine does not set or clear the flags that indicate that
                                 16B9   ;               the data has been truncated at the right side or bottom of the
                                 16B9   ;               screen
                                 16B9   ;
                                 16B9   ; CALLING SEQUENCE:
                                 16B9   ;
                                 16B9   ;       JSB     SET_SCROLL
                                 16B9   ;
                                 16B9   ; CALLED BY:
                                 16B9   ;
                                 16B9   ;       READ_WO_PROMPT
                                 16B9   ;
                                 16B9   ; IMPLICIT INPUTS:
                                 16B9   ;
                                 16B9   ;       Escape sequence stored in CMD_BUF.
                                 16B9   ;
                                 16B9   ; OUTPUTS:
                                 16B9   ;
                                 16B9   ;       V_SCROLL, CLSTR_H_SCROLL, or LOCAL_H_SCROLL updated.
                                 16B9   ;       SCROLL_FLAGS updated.
                                 16B9   ;       SCROLL_DIF updated.
                                 16B9   ;
                                 16B9   ;--
                                 16B9
                                 16B9   SET_SCROLL:
 9B 8F    00002242'EF     91     16B9           CMPB    IOSB+4,#CSI             ; Did the user type an eight-bit escape
                                 16C1                                           ;  sequence?
                  11      12     16C1           BNEQ    10$                     ; Branch if not.
000020DB'EF    5B 8F      90     16C3           MOVB    #^A/[/,CMD_BUF          ; Substitute an open bracket.
     51   000020DB'EF     DE     16CB           MOVAL   CMD_BUF,R1              ; Point to open bracket.
                  07      11     16D2           BRB     20$                     ; Jump to common code.
     51   000020DC'EF     DE     16D4   10$:    MOVAL   CMD_BUF+1,R1            ; Point to open bracket.
  00002247'EF     01      E1     16DB   20$:    BBC     #DVT$V_DCRT,OUT_DEV,-   ; Branch if not DECcrt terminal.
```

```
                      36       16E2            40$                             ;
           445B 8F    61  B1   16E3            CMPW    (R1),#DCRT_RT_ARROW     ; Right arrow?
                               16E8            JEQL    DEC_H_SCROLL            ; Scroll right.
           435B 8F    61  B1   16ED            CMPW    (R1),#DCRT_LFT_ARROW    ; Left arrow?
                               16F2            JEQL    INC_H_SCROLL            ; Scroll left.
                      00  E0   16F7            BBS     #RPT$V_CLSTR,-          ; Branch if we're displaying the
         03 00000000'EF        16F9                    CURRENT_REPORT,30$      ;  cluster report.
                    01C6  31   16FF            BRW     999$                    ; Otherwise, don't allow up or down
                               1702                                            ;  scroll.
           415B 8F    61  B1   1702    30$:    CMPW    (R1),#DCRT_UP_ARROW     ; Up arrow?
                               1707            JEQL    DEC_V_SCROLL            ; Scroll up.
           425B 8F    61  B1   170C            CMPW    (R1),#DCRT_DWN_ARROW    ; Down arrow?
                               1711            JEQL    INC_V_SCROLL            ; Scroll down.
                    01AF  31   1716            BRW     999$                    ; Must be invalid escape sequence.
                               1719
             44 8F    61  91   1719    40$:    CMPB    (R1),#VT52_RT_ARROW     ; Right arrow?
                               171D            JEQL    DEC_H_SCROLL            ; Scroll right.
             43 8F    61  91   1722            CMPB    (R1),#VT52_LFT_ARROW    ; Left arrow?
                               1726            JEQL    INC_H_SCROLL            ; Scroll left.
                      00  E0   172B            BBS     #RPT$V_CLSTR,-          ; Branch if we're displaying the
         03 00000000'EF        172D                    CURRENT_REPORT,50$      ;  cluster report.
                    0192  31   1733            BRW     999$                    ; Otherwise, don't allow up or down
                               1736                                            ;  scroll.
             41 8F    61  91   1736    50$:    CMPB    (R1),#VT52_UP_ARROW     ; Up arrow?
                      0C  13   173A            BEQL    DEC_V_SCROLL            ; Scroll up.
             42 8F    61  91   173C            CMPB    (R1),#VT52_DWN_ARROW    ; Down arrow?
                               1740            JEQL    INC_V_SCROLL            ; Scroll down.
                    0180  31   1745            BRW     999$                    ; Must be invalid escape sequence.
                               1748
                               1748    DEC_V_SCROLL:
               0000045C'EF  D5   1748          TSTL    V_SCROLL                ; Already scrolled all the way down?
                               174E            JEQL    999$                    ; If so, don't scroll.
               00000430'EF  D6   1753          INCL    SCROLL_LINES            ; Increase number of times we are
                               1759                                            ;  allowed to scroll up.
               0000045C'EF  D7   1759          DECL    V_SCROLL                ; Scroll display down.
                      07  12   175F            BNEQ    60$                     ; Branch if we're not all the way down.
                      04  8A   1761            BICB    #SCR$M_MORE_UP,-        ; Otherwise, clear the up flag.
               0000047A'EF     1763                    SCROLL_FLAGS            ;
     03 00002247'EF   01  E0   1768    60$:    BBS     #DVT$V_DCRT,OUT_DEV,70$ ; Branch if terminal is a DECcrt.
                    014E  31   1770            BRW     140$                    ; Otherwise branch to set refresh flag.
                    0266  30   1773    70$:    BSBW    SET_REGION_DATA         ; Set scrolling region to include only
                               1776                                            ;  the data area of the screen.
   00002248'EF   00002254'EF  D0   1776        MOVL    FIRST_DATA_LINE,LINE_NUM; Set the cursor to the first line of
                 0000224C'EF  3F   1781        PUSHAW  COL_NUM                 ;  data.
                 00002248'EF  3F   1787        PUSHAW  LINE_NUM                ;
             00000000'GF  02  FB   178D        CALLS   #2,G^LIB$SET_CURSOR     ;
                               1794            JLBC    R0,999$                 ; Branch on error.
             00000000'GF  00  FB   179A        CALLS   #0,G^LIB$DOWN_SCROLL    ; Scroll down one line.
                               17A1            JLBC    R0,999$                 ; Branch on error.
                    0224  30   17A7            BSBW    SET_REGION_FULL         ; Reset the scrolling region to use the
                               17AA                                            ;  full screen.
00000444'EF   00000444'EF  01  9C  17AA        ROTL    #1,SCROLL_DIF,SCROLL_DIF; Force the top line of data to be
                 00002254'EF  E2   17B6        BBSS    FIRST_DATA_LINE,-       ;  redrawn.
              00 00000444'EF       17BC                SCROLL_DIF,80$          ;
                    0103  31   17C2    80$:    BRW     999$                    ; Finish up.
                               17C5
                               17C5    INC_V_SCROLL:
```

```
           0000007E 8F   D3  17C5          BITL    #CLS$M_ANY_SCS,-          ; Any SCS fields being displayed?
           00000001'EF       17CB                  CLASS_SELECT              ;
                             17D0          JEQL    999$                      ; Branch if not.
                             17D5  ;
                             17D5  ; Since we don't really know how many lines might be displayed in a given
                             17D5  ; report, we don't really know how many times we can up-scroll.  Because of
                             17D5  ; this, we limit the number of times the user can hit the up arrow before
                             17D5  ; we get a chance to redraw the screen and see how many more lines we have.
                             17D5  ;
           00000430'EF   D5  17D5          TSTL    SCROLL_LINES              ; Any more lines to be scrolled?
                             17DB          JEQL    999$                      ; Branch if not.
           00000430'EF   D7  17E0          DECL    SCROLL_LINES              ; Now there's one less.
           0000045C'EF   D6  17E6          INCL    V_SCROLL                  ; Scroll display up.
        58 00002247'EF  01  E1  17EC       BBC     #DVT$V_DCRT,OUT_DEV,90$   ; Branch if terminal isn't a DECcrt.
                 01E5   30  17F4          BSBW    SET_REGION_DATA           ; Set scrolling region to include only
                             17F7                                            ;  the data area of the screen.
00002248'EF  00002258'EF  D0  17F7        MOVL    LAST_DATA_LINE,LINE_NUM   ; Set the cursor to the last line of
           0000224C'EF   3F  1802          PUSHAW  COL_NUM                   ;  data.
           00002248'EF   3F  1808          PUSHAW  LINE_NUM                  ;
        00000000'GF  02  FB  180E          CALLS   #2,G^LIB$SET_CURSOR       ;
                             1815          JLBC    R0,999$                   ; Branch on error.
        00000000'GF  00  FB  181B          CALLS   #0,G^LIB$UP_SCROLL        ; Scroll up one line.
                             1822          JLBC    R0,999$                   ; Branch on error.
                 01A3   30  1828          BSBW    SET_REGION_FULL           ; Reset the scrolling region to use the
                             182B                                            ;  full screen.
                   03   E1  182B          BBC     #SCR$V_MORE_DWN,-         ; Branch if the bottom line doesn't
        19 0000047A'EF       182D                  SCROLL_FLAGS,90$          ;  need to be drawn.
00000444'EF    FF 8F  9C  1833            ROTL    #-1,SCROLL_DIF,-          ; Force the last data line to be
           00000444'EF       183B                  SCROLL_DIF                ;  drawn.
           00002258'EF   E2  1840          BBSS    LAST_DATA_LINE,-          ;
        00 00000444'EF       1846                  SCROLL_DIF,90$            ;
                   14   88  184C  90$:    BISB    #SCR$M_MORE_UP!SCR$M_MORE,-
           0000047A'EF       184E                  SCROLL_FLAGS              ; Indicate screen can be scrolled down.
        66 00002247'EF  01  E1  1853       BBC     #DVT$V_DCRT,OUT_DEV,140$; Branch if terminal isn't a DECcrt.
                   6B   11  185B          BRB     999$                      ; Otherwise, don't set the refresh flag.
                             185D
                             185D  INC_H_SCROLL:
           00000425'EF   95  185D          TSTB    NUM_FIELDS                ; Any fields to be displayed?
                   63   13  1863          BEQL    999$                      ; Branch if not.
                   00   E1  1865          BBC     #RPT$V_CLSTR,-            ; Branch if we're not displaying the
        08 00000000'EF       1867                  CURRENT_REPORT,100$       ;  cluster report.
           00000464'EF   D6  186D          INCL    CLSTR_H_SCROLL            ; Otherwise scroll the cluster report
                             1873                                            ;  left.
                   06   11  1873          BRB     110$                      ; Branch to common code.
           00000468'EF   D6  1875  100$:   INCL    LOCAL_H_SCROLL            ; Scroll the local report left.
           00000425'EF   97  187B  110$:   DECB    NUM_FIELDS                ; Decrease number of times we can scroll
                             1881                                            ;  left.
                   12   88  1881          BISB    #SCR$M_MORE_LFT!SCR$M_MORE,-
           0000047A'EF       1883                  SCROLL_FLAGS              ; Indicate screen can be scrolled
                             1888                                            ;  right.
                   37   11  1888          BRB     140$                      ; Done.
                             188A
                             188A  DEC_H_SCROLL:
                   00   E1  188A          BBC     #RPT$V_CLSTR,-            ; Branch if we're not displaying the
        12 00000000'EF       188C                  CURRENT_REPORT,120$       ;  cluster report.
           00000464'EF   D5  1892          TSTL    CLSTR_H_SCROLL            ; Is the cluster report scrolled all
                             1898                                            ;  the way to the right?
```

```
                       2E  13  1898  5238            BEQL    999$                    ; Don't scroll anymore if so.
               00000464'EF  D7  189A  5239            DECL    CLSTR_H_SCROLL          ; Otherwise scroll the cluster report
                                18A0  5240                                            ;  right.
                       19  12  18A0  5241            BNEQ    135$                    ; Branch if the cluster report isn't
                                18A2  5242                                            ;  scrolled all the way to the right.
                       10  11  18A2  5243            BRB     130$                    ; Branch to common code.
                                18A4  5244
               00000468'EF  D5  18A4  5245 120$:      TSTL    LOCAL_H_SCROLL          ; Is the local report scrolled all the
                                18AA  5246                                            ;  way to the right?
                       1C  13  18AA  5247            BEQL    999$                    ; Don't scroll anymore if so.
               00000468'EF  D7  18AC  5248            DECL    LOCAL_H_SCROLL          ; Otherwise, scroll the local report
                                18B2  5249                                            ;  right.
                       07  12  18B2  5250            BNEQ    135$                    ; Branch if the local report isn't
                                18B4  5251                                            ;  scrolled all the way to the right.
                       02  8A  18B4  5252 130$:      BICB    #SCR$M_MORE_LFT,-       ; Clear left flag.
               0000047A'EF      18B6  5253                    SCROLL_FLAGS            ;
               00000425'EF  96  18BB  5254 135$:      INCB    NUM_FIELDS              ; Increase number of times we can scroll
                                18C1  5255                                            ;  left.
          00000476'EF  08  C8  18C1  5256 140$:      BISL    #MF$M_RIP,MISC_FLAGS    ; Set flag to indicate that the screen
                                18C8  5257                                            ;  should be refreshed.
  00000476'EF    00000100 8F  C8  18C8  5258 999$:      BISL    #MF$M_SCROLL,MISC_FLAGS ; Indicate we just scrolled.
                            05  18D3  5259            RSB
                                18D4  5260
                                18D4  5261            .DSABL  LSB
                                18D4  5262
                                18D4  5263
```

```
                                18D4         .SBTTL          -       CHECK_MORE routine.
                                18D4  ;++
                                18D4  ;
                                18D4  ; FUNCTIONAL DESCRIPTION:
                                18D4  ;
                                18D4  ;      This routine is used to output the "MORE" scrolling signals at the
                                18D4  ;      bottom of the display.  It first calls the routine which outputs
                                18D4  ;      broadcast messages (giving the messages priority over the scrolling
                                18D4  ;      signals).  It then checks to see if any scroll flags are set.  If
                                18D4  ;      there is a scroll flag set it then checks to see if any information has
                                18D4  ;      been output on either of the command lines (the bottom two lines of the
                                18D4  ;      display - the top line has priority).  If there is no information
                                18D4  ;      indicated on one of the lines it adds the "MORE" scrolling signals to
                                18D4  ;      the screen buffer.
                                18D4  ;
                                18D4  ; CALLING SEQUENCE:
                                18D4  ;
                                18D4  ;      JSB CHECK_MORE
                                18D4  ;
                                18D4  ; IMPLICIT INPUTS:
                                18D4  ;
                                18D4  ;      TOP_CMD_LINE and BOT_CMD_LINE indicate the type of data currently being
                                18D4  ;              displayed in the command scroll region.
                                18D4  ;
                                18D4  ; IMPLICIT OUTPUTS:
                                18D4  ;
                                18D4  ;      Scrolling signals output to the screen buffer if no other data is being
                                18D4  ;      displayed in the command scroll region.
                                18D4  ;
                                18D4  ; RETURN STATUS:
                                18D4  ;
                                18D4  ;      SS$_NORMAL or failed status returned in R0.
                                18D4  ;--
                                18D4
                                18D4  CHECK_MORE:
                  FD43    30    18D4         BSBW    OUT_MSG                 ; First see if there are any broadcast
                                18D7                                         ;  messages to output.
             04     00    EC    18D7         CMPV    #SCR$V_MORE_RT,#4,-     ; Has any data been scrolled off the
     00     0000047A'EF         18DA                 SCROLL_FLAGS,#0         ;  screen or truncated?
                                18E0         JEQL    99$                     ; Branch if not.
            00000471'EF   95    18E5         TSTB    BOT_CMD_LINE            ; Is the bottom command line blank?
                                18EB         JNEQ    99$                     ; Branch if not.
00002248'EF 00002260'EF   D0    18F0         MOVL    LAST_LINE,LINE_NUM      ; Output on the bottom command line.
  00002222'EF    0050 8F  B1    18FB         CMPW    #NARROW,SCREEN_WIDTH    ; Is the terminal width 80 columns?
                     14   13    1904         BEQL    5$                      ; Branch if it is.
            0000224C'EF   3F    1906         PUSHAW  COL_NUM                 ; Output "                More   ".
            00002248'EF   3F    190C         PUSHAW  LINE_NUM
            00000090'EF   3F    1912         PUSHAW  MORE_TABS
                     12   11    1918         BRB     7$
            0000224C'EF   3F    191A  5$:    PUSHAW  COL_NUM                 ; Output 'More  ".
            00002248'EF   3F    1920         PUSHAW  LINE_NUM
            00000078'EF   3F    1926         PUSHAW  MORE_DSCR
       00000000'GF   03   FB    192C  7$:    CALLS   #3,G^LIB$PUT_SCREEN
                                1933         JLBC    R0,99$                  ; Branch on error.
                     04   93    1939         BITB    #SCR$M_MORE_UP,-        ; Has the screen been scrolled up?
            0000047A'EF         193B                 SCROLL_FLAGS
                     12   13    1940         BEQL    10$                     ; Branch if not.
```

```
 000000A7'EF   3F  1942  5322          PUSHAW  MORE_UP                 ; Output "^    ".
 00000000'GF   01  FB  1948  5323      CALLS   #1,G^LIB$PUT_SCREEN     ;
       7B 50   E9  194F  5324          BLBC    R0,99$                  ; Branch on error.
          10   11  1952  5325          BRB     20$                     ; Jump to common code.
 000000D3'EF   3F  1954  5326 10$:     PUSHAW  NO_MORE                 ; Output "     ".
 00000000'GF   01  FB  195A  5327      CALLS   #1,G^LIB$PUT_SCREEN     ;
       69 50   E9  1961  5328          BLBC    R0,99$                  ; Branch on error.
          08   93  1964  5329 20$:     BITB    #SCR$M_MORE_DWN,-       ; Are there more lines of data than can
 0000047A'EF       1966  5330                  SCROLL_FLAGS            ;  be displayed?
          12   13  196B  5331          BEQL    30$                     ; Branch if not.
 000000B2'EF   3F  196D  5332          PUSHAW  MORE_DWN                ; Output "v    ".
 00000000'GF   01  FB  1973  5333      CALLS   #1,G^LIB$PUT_SCREEN     ;
       50 50   E9  197A  5334          BLBC    R0,99$                  ; Branch on error.
          0D   11  197D  5335          BRB     50$                     ; Jump to common code.
 000000D3'EF   3F  197F  5336 30$:     PUSHAW  NO_MORE                 ; Output "     ".
 00000000'GF   01  FB  1985  5337 40$: CALLS   #1,G^LIB$PUT_SCREEN     ;
          02   93  198C  5338 50$:     BITB    #SCR$M_MORE_LFT,-       ; Has the screen been scrolled right?
 0000047A'EF       198E  5339                  SCROLL_FLAGS            ;
          12   13  1993  5340          BEQL    60$                     ; Branch if not.
 000000BD'EF   3F  1995  5341          PUSHAW  MORE_LFT                ; Output "<    ".
 00000000'GF   01  FB  199B  5342      CALLS   #1,G^LIB$PUT_SCREEN     ;
       28 50   E9  19A2  5343          BLBC    R0,99$                  ; Branch on error.
          10   11  19A5  5344          BRB     70$                     ; Jump to common code.
 000000D3'EF   3F  19A7  5345 60$:     PUSHAW  NO_MORE                 ; Output "     ".
 00000000'GF   01  FB  19AD  5346      CALLS   #1,G^LIB$PUT_SCREEN     ;
       16 50   E9  19B4  5347          BLBC    R0,99$                  ; Branch on error.
          01   93  19B7  5348 70$:     BITB    #SCR$M_MORE_RT,-        ; Has the data been truncated?
 0000047A'EF       19B9  5349                  SCROLL_FLAGS            ;
          0D   13  19BE  5350          BEQL    99$                     ; Branch if not.
 000000C8'EF   3F  19C0  5351          PUSHAW  MORE_RT                 ; Output ">".
 00000000'GF   01  FB  19C6  5352      CALLS   #1,G^LIB$PUT_SCREEN     ;
               05  19CD  5353 99$:     RSB
                   19CE  5354
                   19CE  5355
```

```
                                   19CE  5357          .SBTTL          -       Set scrolling region routines.
                                   19CE  5358          .ENABL  LSB
                                   19CE  5359  ;++
                                   19CE  5360  ;
                                   19CE  5361  ; FUNCTIONAL DESCRIPTION:
                                   19CE  5362  ;
                                   19CE  5363  ;       The followin four routines set the scrolling regions of the terminal.
                                   19CE  5364  ;       The four regions are defined as follows:
                                   19CE  5365  ;
                                   19CE  5366  ;       FULL            The full screen
                                   19CE  5367  ;       DATA            The line imediately following the header up to the
                                   19CE  5368  ;                       command region.
                                   19CE  5369  ;       COMMAND         The last two lines of the screen.
                                   19CE  5370  ;       HELP            The line immediately following the header through the
                                   19CE  5371  ;                       last line of the screen.
                                   19CE  5372  ;
                                   19CE  5373  ; CALLING SEQUENCE:
                                   19CE  5374  ;
                                   19CE  5375  ;       JSB     SET_REGION_FULL
                                   19CE  5376  ;       JSB     SET_REGION_COMMAND
                                   19CE  5377  ;       JSB     SET_REGION_DATA
                                   19CE  5378  ;       JSB     SET_REGION_HELP
                                   19CE  5379  ;
                                   19CE  5380  ; CALLED BY:
                                   19CE  5381  ;
                                   19CE  5382  ;       READ_W_PROMPT
                                   19CE  5383  ;       DEC_V_SCROLL
                                   19CE  5384  ;       INC_V_SCROLL
                                   19CE  5385  ;       SCROLL_TWO
                                   19CE  5386  ;       GET_HELP
                                   19CE  5387  ;
                                   19CE  5388  ; RETURN STATUS:
                                   19CE  5389  ;
                                   19CE  5390  ;       SS$_NORMAL or failed status returned in R0.
                                   19CE  5391  ;--
                                   19CE  5392
                                   19CE  5393  SET_REGION_FULL:
             00002250'EF     3F    19CE  5394          PUSHAW  FIRST_LINE                      ; Reset the scrolling region to use
             00002260'EF     3F    19D4  5395          PUSHAW  LAST_LINE                       ;  the full screen.
                      3A     11    19DA  5396          BRB     10$                             ; Branch to common code.
                                   19DC  5397
                                   19DC  5398  SET_REGION_DATA:
             00002258'EF     3F    19DC  5399          PUSHAW  LAST_DATA_LINE                  ; Set the scrolling region to use only
             00002254'EF     3F    19E2  5400          PUSHAW  FIRST_DATA_LINE                 ;  the data area of the display.
                      2C     11    19E8  5401          BRB     10$                             ; Branch to common code.
                                   19EA  5402
                                   19EA  5403  SET_REGION_COMMAND:
00002248'EF  0000225C'EF     D0    19EA  5404          MOVL    CMD_LINE,LINE_NUM               ; Set the scrolling region to the last
             00002260'EF     3F    19F5  5405          PUSHAW  LAST_LINE                       ;  two lines of the display.
             00002248'EF     3F    19FB  5406          PUSHAW  LINE_NUM                        ;
                      13     11    1A01  5407          BRB     10$                             ; Branch to common code.
                                   1A03  5408
                                   1A03  5409  SET_REGION_HELP:
          00002248'EF   07   D0    1A03  5410          MOVL    #HEADER_END,LINE_NUM            ; Set the scrolling region to the area
             00002260'EF     3F    1A0A  5411          PUSHAW  LAST_LINE                       ;  from the line following the header
             00002248'EF     3F    1A10  5412          PUSHAW  LINE_NUM                        ;  to the last line of the display.
                                   1A16  5413
```

```
          00000000'GF   02  FB  1A16  5414 10$:   CALLS   #2,G^LIB$SET_SCROLL           ;
                            05  1A1D  5415        RSB                                   ;
                                1A1E  5416
                                1A1E  5417        .DSABL  LSB
                                1A1E  5418
```

```
          1A1E                 .SBTTL  AST routines.
          1A1E         ;++
          1A1E         ;
          1A1E         ; FUNCTIONAL DESCRIPTION:
          1A1E         ;
          1A1E         ;         This routine is entered when data has been placed in the mailbox
          1A1E         ;         associated with the terminal.  This data may be from a broadcast
          1A1E         ;         message, unsolicited input from the terminal, or terminal hangup.
          1A1E         ;         This routine simply sets an event flag and return so that we do
          1A1E         ;         not remain at AST delivery very long.
          1A1E         ;
          1A1E         ; OUPUTS:
          1A1E         ;
          1A1E         ;         AST event flag set.
          1A1E         ;
          1A1E         ;--
          1A1E
    0000  1A1E         .ENTRY    AST_DEL,^M<>
          1A20                   $SETEF_S -                      ; Set the AST event flag.
          1A20                           EFN=#EVF$V_AST_FLAG     ;
      04  1A29                   RET
          1A2A
          1A2A
```

```
                  1A2A  ;++
                  1A2A  ;
                  1A2A  ; FUNCTIONAL DESCRIPTION:
                  1A2A  ;
                  1A2A  ;        This routine is entered when the user types control-C on the terminal.
                  1A2A  ;        It simply sets the exit flag to indicate that the user has requested
                  1A2A  ;        that the program exit and then transfers control to the exit routine.
                  1A2A  ;
                  1A2A  ; OUTPUTS:
                  1A2A  ;
                  1A2A  ;        Exit flag set at MISC_FLAGS.
                  1A2A  ;
                  1A2A  ;--
                  1A2A
            0000  1A2A  .ENTRY   CNTRLC_AST_DEL,^M<>
                  1A2C
    50   01   D0  1A2C           MOVL    #SS$_NORMAL,R0           ; Indicate normal exit.
                  1A2F           $EXIT_S                          ; Exit.
              04  1A38           RET
                  1A39
                  1A39
```

```
               1A39  ;++
               1A39  ;
               1A39  ; FUNCTIONAL DESCRIPTION:
               1A39  ;
               1A39  ;       This routine queues a read over the mailbox channel which has been
               1A39  ;       associated with the terminal and then enables the mailbox.
               1A39  ;
               1A39  ; CALLING SEQUENCE:
               1A39  ;
               1A39  ;       JSB     SET_AST
               1A39  ;
               1A39  ; CALLED BY:
               1A39  ;
               1A39  ;       INIT
               1A39  ;       MAIN_CONT
               1A39  ;
               1A39  ; RETURN STATUS:
               1A39  ;
               1A39  ;       SS$_NORMAL returned in R0.
               1A39  ;--
               1A39
               1A39  SET_AST:
               1A39          $QIO_S -                                ; Set the AST.
               1A39                  CHAN=MBXCHAN,-
               1A39                  FUNC=#IO$_READVBLK,-
               1A39                  IOSB=IOSB,-
               1A39                  ASTADR=AST_DEL,-
               1A39                  P1=MBX_BUF,-
               1A39                  P2=#MBX_BUF_LEN
   2C 50   E9  1A69          BLBC    R0,99$                          ; Branch on error.
               1A6C
               1A6C          $QIO_S -                                ; Enable mailbox.
               1A6C                  CHAN=IN_CHAN,-
               1A6C                  FUNC=#IO$_WRITEVBLK!IO$M_ENABLMBX,-
               1A6C                  IOSB=IOSB,-
               1A6C                  P1=CMD_BUF,-
               1A6C                  P2=#0
               1A95
50   01    D0  1A95          MOVL    #SS$_NORMAL,R0
           05  1A98  99$:    RSB
               1A99
               1A99
               1A99          .ENABL  LSB
               1A99
```

```
1A99  ;++
1A99  ;
1A99  ; FUNCTIONAL DESCRIPTION:
1A99  ;
1A99  ;       This routine is used to erase the screen from the end of the report up
1A99  ;       to the last two lines of the display.  It also erases the line where
1A99  ;       the scrolling signals appear if they are being output.  This insures
1A99  ;       that the scrolling signals always reflect the correct status and also
1A99  ;       produces a blinking effect.
1A99  ;
1A99  ;       This routine also determines if the bottom two lines of the display
1A99  ;       should be scrolled.  This scrolling is done only if there is additional
1A99  ;       information on the bottom command line or if there is no information
1A99  ;       on the bottom command line and the top command line does not contain
1A99  ;       a broadcast message. The results (listed in priority) appear as follows:
1A99  ;
1A99  ;       1) Command input and error messages are scrolled in the command region
1A99  ;          on each refresh until they disappear from the display.
1A99  ;
1A99  ;       2) Broadcast messages are scrolled in the command region until the last
1A99  ;          broadcast message is on the top line in the command region at which
1A99  ;          point it is left there until the user takes action to remove it.
1A99  ;
1A99  ;       3) Scrolling signals are erased and are restored after the next update
1A99  ;          on the first blank line in the command region.
1A99  ;
1A99  ; CALLING SEQUENCE:
1A99  ;
1A99  ;       JSB     ERASE_PAGE
1A99  ;
1A99  ; CALLED BY:
1A99  ;
1A99  ;       DISPLAY_END
1A99  ;
1A99  ; INPUTS:
1A99  ;
1A99  ;       Number of the line following the report is stored in LINE_NUM.
1A99  ;
1A99  ; IMPLICIT INPUTS:
1A99  ;
1A99  ;       SCROLL_FLAGS    have been set to indicate the direction the screen can
1A99  ;                       be scrolled.
1A99  ;       TOP_CMD_LINE and BOT_CMD_LINE indicate the type of data currently being
1A99  ;               displayed in the command scroll region.
1A99  ;
1A99  ; IMPLICIT OUTPUTS:
1A99  ;
1A99  ;       Bottom two lines of display are scrolled.
1A99  ;       TOP_CMD_LINE and BOT_CMD_LINE are updated.
1A99  ;
1A99  ; RETURN STATUS:
1A99  ;
1A99  ;       SS$_NORMAL or failed status returned in R0.
1A99  ;
1A99  ; REGISTER USAGE:
1A99  ;
1A99  ;       R2 -    Used to determine when all the lines have been erased.
```

```
                                  1A99  ;--
                                  1A99
                                  1A99  ERASE_PAGE:
              50     01     D0    1A99             MOVL    #SS$_NORMAL,R0           ; Assume success.
    0D 00000476'EF   03     E0    1A9C             BBS     #MF$V_RIP,MISC_FLAGS,5$  ; Branch if refresh in progress.
       00000428'EF          D1    1AA4             CMPL    NUM_LINES,-              ; Do we have the same number of lines
       0000042C'EF                1AAA                     PRIOR_NUM_LINES          ;  of data as last pass?
                     1F     13    1AAF             BEQL    14$                      ; If so skip the erase.
                     04     BB    1AB1  5$:        PUSHR   #^M<R2>
        52    00002258'EF   D0    1AB3             MOVL    LAST_DATA_LINE,R2        ; Erase all the remaining data.
        52    00002248'EF   D1    1ABA  10$:       CMPL    LINE_NUM,R2              ; Any lines to erase?
                     0B     14    1AC1             BGTR    11$                      ; Branch if not.
                   093C     30    1AC3             BSBW    ERASE_LINE               ; Erase one line.
                                  1AC6             JLBC    R0,99$                   ; Branch on error.
                     EC     11    1ACC             BRB     10$                      ; Loop.
                     04     BA    1ACE  11$:       POPR    #^M<R2>
                                  1AD0
                                  1AD0  ;
                                  1AD0  ; If we are outputing "more" on the bottom line then erase the line so that it
                                  1AD0  ; reflects the correct status.
                                  1AD0  ;
                                  1AD0
       00000428'EF          D0    1AD0  14$:       MOVL    NUM_LINES,-              ; Save number of lines for next pass.
       0000042C'EF                1AD6                     PRIOR_NUM_LINES          ;
    00000428'EF      07     C3    1ADB             SUBL3   #7,NUM_LINES,-           ; Save number of times we can up scroll.
       00000430'EF                1AE2                     SCROLL_LINES             ;
                     00     E1    1AE7             BBC     #CLS$V_CLUB,-            ; Branch if not displaying the CLUSTER
    07 00000001'EF                1AE9                     CLASS_SELECT,15$         ;  class.
    00000430'EF      07     C2    1AEF             SUBL    #7,SCROLL_LINES          ; Otherwise, adjust number of lines that
                                  1AF6                                              ;  can be scrolled.
                     04     E1    1AF6  15$:       BBC     #SCR$V_MORE,-            ; Branch if none of the scroll flags
    2E 0000047A'EF                1AF8                     SCROLL_FLAGS,20$         ;  have been set.
              04     00     EC    1AFE             CMPV    #SCR$V_MORE_RT,#4,-      ; Are any of the scroll flags still
        00    0000047A'EF         1B01                     SCROLL_FLAGS,#0          ;  set?
                     07     12    1B07             BNEQ    16$                      ; Branch if so.
    0000047A'EF      10     8A    1B09             BICB    #SCR$M_MORE,SCROLL_FLAGS; Otherwise clear the summary flag.
       00000471'EF          95    1B10  16$:       TSTB    BOT_CMD_LINE             ; Is the bottom command line blank?
                     14     12    1B16             BNEQ    20$                      ; Branch if not.
00002248'EF   00002260'EF   D0    1B18             MOVL    LAST_LINE,LINE_NUM       ; Get the number of the bottom command
                                  1B23                                              ;   line.
                   08DC     30    1B23             BSBW    ERASE_LINE               ; Erase the line that we are outputing
                                  1B26                                              ;  "more" on.
                                  1B26             JLBC    R0,99$                   ; Branch on error.
       00000458'EF          D5    1B2C  20$:       TSTL    SCROLL_DELAY             ; Time to scroll the bottom two lines?
                     09     13    1B32             BEQL    30$                      ; Branch if so.
       00000458'EF          D7    1B34             DECL    SCROLL_DELAY             ; Count down the intervals before it's
                                  1B3A                                              ;  time to scroll.
                   0108     31    1B3A             BRW     99$                      ; Done for now.
       00000471'EF          95    1B3D  30$:       TSTB    BOT_CMD_LINE             ; Any information on the bottom command
                                  1B43                                              ;  line?
                                  1B43             JNEQ    SCROLL_TWO               ; Branch if so.
       00000470'EF          95    1B48             TSTB    TOP_CMD_LINE             ; Any information on the top command
                                  1B4E                                              ;  line?
                                  1B4E             JEQL    99$                      ; If not, no need to scroll.
                     01     93    1B53             BITB    #CL$M_BRDCST,-           ; Is there a broadcast message on the
       00000470'EF                1B55                     TOP_CMD_LINE             ;  top command line?
                                  1B5A             JNEQ    99$                      ; If so, don't scroll.
```

```
                                    1B5F  5625
                                    1B5F  5626 SCROLL_TWO:
         3E 00002247'EF     01   E1 1B5F  5627         BBC     #DVT$V_DCRT,OUT_DEV,40$ ; Branch if terminal is a VT52 type.
                      FE7F CF    16 1B67  5628         JSB     SET_REGION_COMMAND      ; Establish a scrolling region.
                                    1B6B  5629         JLBC    R0,99$                  ; Branch on error.
  00002248'EF     00002260'EF    D0 1B71  5630         MOVL    LAST_LINE,LINE_NUM      ; Set the cursor to the last line.
                  0000224C'EF    3F 1B7C  5631         PUSHAW  COL_NUM                 ;
                  00002248'EF    3F 1B82  5632         PUSHAW  LINE_NUM                ;
            00000000'GF     02   FB 1B88  5633         CALLS   #2,G^LIB$SET_CURSOR     ;
                                    1B8F  5634         JLBC    R0,99$                  ; Branch on error.
            00000000'GF     00   FB 1B95  5635         CALLS   #0,G^LIB$UP_SCROLL      ; Scroll up one line.
                                    1B9C  5636         JLBC    R0,99$                  ; Branch on error.
                          008F   31 1BA2  5637         BRW     80$                     ; Jump to common code.
         03 00002247'EF     02   E1 1BA5  5638 40$:    BBC     #DVT$V_HARD,OUT_DEV,43$ ; Branch if output device isn't
                                    1BAD  5639                                         ;  hardcopy.
                          0084   31 1BAD  5640         BRW     80$                     ; Otherwise, jump to common code.
            00002247'EF     03   E0 1BB0  5641 43$:    BBS     #DVT$V_UNKNOWN,OUT_DEV,-; Branch if terminal type is unknown.
                            7C      1BB7  5642                 80$
            00000471'EF          95 1BB8  5643         TSTB    BOT_CMD_LINE            ; Any information on the bottom command
                                    1BBE  5644                                         ;  line?
                            4B   13 1BBE  5645         BEQL    50$                     ; Branch if not.
  00002248'EF     0000225C'EF    D0 1BC0  5646         MOVL    CMD_LINE,LINE_NUM       ; Set the cursor to the top command
                  0000224C'EF    3F 1BCB  5647         PUSHAW  COL_NUM                 ;  line.
                  00002248'EF    3F 1BD1  5648         PUSHAW  LINE_NUM                ;
            00000000'GF     02   FB 1BD7  5649         CALLS   #2,G^LIB$SET_CURSOR     ;
                         64 50   E9 1BDE  5650         BLBC    R0,99$                  ; Branch on error.
                            01   93 1BE1  5651         BITB    #CL$M_BRDCST,-          ; Is there a broadcast message on the
            00000471'EF             1BE3  5652                 BOT_CMD_LINE            ;  on the bottom command line.
                            12   13 1BE8  5653         BEQL    45$                     ; If not, must be error message.
            00001DF2'EF          3F 1BEA  5654         PUSHAW  LAST_BRDCST_DSCR        ; Output the last broadcast message on
            00000000'GF     01   FB 1BF0  5655         CALLS   #1,G^LIB$PUT_SCREEN     ;   the top command line.
                         4B 50   E9 1BF7  5656         BLBC    R0,99$                  ; Branch on error.
                            03   11 1BFA  5657         BRB     46$                     ; Jump to common code.
                          F95E   30 1BFC  5658 45$:    BSBW    SIGNAL                  ; Output the last error message on the
                                    1BFF  5659                                         ;  top command line.
            00000000'GF     00   FB 1BFF  5660 46$:    CALLS   #0,G^LIB$ERASE_PAGE     ; Erase the bottom line.
                         3C 50   E9 1C06  5661         BLBC    R0,99$                  ; Branch on error.
                            29   11 1C09  5662         BRB     80$                     ; Jump to common code.
            00000470'EF          95 1C0B  5663 50$:    TSTB    TOP_CMD_LINE            ; Any information on the top command
                                    1C11  5664                                         ;  line?
                            32   13 1C11  5665         BEQL    99$                     ; Branch if not.
  00002248'EF     0000225C'EF    D0 1C13  5666         MOVL    CMD_LINE,LINE_NUM       ; Erase the top command line.
                  0000224C'EF    3F 1C1E  5667         PUSHAW  COL_NUM                 ;
                  00002248'EF    3F 1C24  5668         PUSHAW  LINE_NUM                ;
            00000000'GF     02   FB 1C2A  5669         CALLS   #2,G^LIB$ERASE_PAGE     ;
                         11 50   E9 1C31  5670         BLBC    R0,99$                  ; Branch on error.
            00000471'EF          90 1C34  5671 80$:    MOVB    BOT_CMD_LINE,-          ; Bottom command line information now
            00000470'EF             1C3A  5672                 TOP_CMD_LINE            ;  on the command line.
            00000471'EF          94 1C3F  5673         CLRB    BOT_CMD_LINE            ; No information on bottom command
                                    1C45  5674                                         ;  line.
                                 05 1C45  5675 99$:    RSB
                                    1C46  5676
                                    1C46  5677         .DSABL  LSB
                                    1C46  5678
                                    1C46  5679
```

```
                          1C46          .SBTTL  Display header generation.
                          1C46  ;++
                          1C46  ;
                          1C46  ; FUNCTIONAL DESCRIPTION:
                          1C46  ;
                          1C46  ;       This routine combines the class titles and the field titles to form the
                          1C46  ;       display header.  It also adds the necessary strings to draw the box
                          1C46  ;       around the header.  It outputs a message identifying the report and the
                          1C46  ;       displays the time in the upper right hand corner.
                          1C46  ;
                          1C46  ; CALLING SEQUENCE:
                          1C46  ;
                          1C46  ;       JSB     GET_HEADER
                          1C46  ;
                          1C46  ;
                          1C46  ; CALLED BY:
                          1C46  ;
                          1C46  ;       DISPLAY_CLUB
                          1C46  ;       DISPLAY_SCS
                          1C46  ;       DISPLAY_PTP
                          1C46  ;
                          1C46  ; OUTPUTS:
                          1C46  ;
                          1C46  ;          The entire header is stored in the screen buffer for
                          1C46  ;       output using LIB$PUT_BUFFER.
                          1C46  ;
                          1C46  ; RETURN STATUS:
                          1C46  ;
                          1C46  ;       R0 =    SS$_Normal or failed status from LIB$PUT_SCREEN.
                          1C46  ;
                          1C46  ; REGISTER USAGE:
                          1C46  ;
                          1C46  ;       R7  -   Used to in formating the system id.
                          1C46  ;       R10 -   Used to in formating the system id.
                          1C46  ;       R11 -   Pointer to descriptor for data to be stored in the screen
                          1C46  ;               buffer.
                          1C46  ;
                          1C46  ;--
                          1C46
                          1C46 GET_HEADER:
           0800 8F   BB   1C46          PUSHR   #^M<R11>
  2B 00000476'EF  03  E0  1C4A          BBS     #MF$V_RIP,MISC_FLAGS,30$; Branch if refresh in progress.
                          1C52  ;
                          1C52  ; When not doing a refresh, just output the time in the upper right hand corner.
                          1C52  ;
                          1C52  ; If the CLUSTER class is being display along with any SCS class, then display
                          1C52  ; the time when GET_HEADER is called for the CLUSTER class only, so that we
                          1C52  ; don't update the time twice.
                          1C52  ;
                          1C52
              00     E1   1C52          BBC     #RPT$V_CLSTR,-          ; Branch if displaying the local report.
     10 00000000'EF       1C54                  CURRENT_REPORT,10$      ;
              00     E0   1C5A          BBS     #CLS$V_CLUB,-           ; Branch if this header is for the
     08 00000001'EF       1C5C                  CLASS_SELECT,10$        ;  CLUSTER class.
              00     E0   1C62          BBS     #CLS$V_CLUB,-           ; Branch if we have already output
     09 000021FC'EF       1C64                  SAVED_CLASSES,20$       ;  the time.
            01F5     30   1C6A 10$:     BSBW    OUTPUT_TIME             ; Output the time.
```

```
         00002248'EF  D6  1C6D        INCL   LINE_NUM
 00002248'EF      05  C0  1C73  20$:  ADDL   #5,LINE_NUM                ; Skip past header info.
                0112  31  1C7A        BRW    998$                       ; And we're done.
                          1C7D
                04C4  30  1C7D  30$:  BSBW   GET_TITLES                 ; Build the title line in TITLE_BUF.
                          1C80        JLBC   R0,999$                    ; Branch on error.
      01 000021C4'EF  D1  1C86        CMPL   HEADER_SIZE,#1             ; Are there any classes or fields being
                          1C8D                                          ;  displayed?
                  06  12  1C8D        BNEQ   40$                        ; Branch if so.
         000021C4'EF  D4  1C8F        CLRL   HEADER_SIZE                ; Otherwise, make sure the first field
                          1C95                                          ;  delimiter doesn't get displayed.
                0608  30  1C95  40$:  BSBW   GET_CLASSES                ; Build the class line in CLASS_BUF.
                          1C98        JLBC   R0,999$                    ; Branch on error.
   05 00002247'EF  01  E1  1C9E        BBC    #DVT$V_DCRT,OUT_DEV,50$    ; Branch if terminal isn't a DECcrt.
                02F4  30  1CA6        BSBW   BUILD_DECCRT_BOX           ; Get the four lines of data that make
                          1CA9                                          ;  up the boxes around the header.
                  03  11  1CA9        BRB    60$                        ; Jump to common code.
                0296  30  1CAB  50$:  BSBW   BUILD_VT52_BOX             ; Build the line for above the classes
                          1CAE                                          ;  in CLASS_BOX_BUF and the line for
                          1CAE                                          ;  above and below the field titles in
                          1CAE                                          ;  FIELD_BOX_BUF.
                          1CAE
                  00  E1  1CAE  60$:  BBC    #RPT$V_CLSTR,-             ; Branch if displaying the local report.
      7B 00000000'EF      1CB0               CURRENT_REPORT,90$         ;
                  00  E1  1CB6        BBC    #CLS$V_CLUB,-              ; Branch if this header isn't for the
      31 00000001'EF      1CB8               CLASS_SELECT,70$           ;  CLUSTER class.
         000021C4'EF  D0  1CBE        MOVL   HEADER_SIZE,-              ; Save size of CLUB header.
         000021C8'EF      1CC4               CLUB_HEADER_SIZE           ;
         00000425'EF  90  1CC9        MOVB   NUM_FIELDS,-               ; Save the number of CLUB fields being
         00000427'EF      1CCF               CLUB_NUM_FIELDS            ;  displayed.
                00C0  30  1CD4        BSBW   OUTPUT_DDM                 ; Output the display descriptor message.
         0000007E 8F  D3  1CD7        BITL   #CLS$M_ANY_SCS,-           ; Are any SCS fields being displayed in
         000021FC'EF      1CDD               SAVED_CLASSES              ;  this report?
                  03  12  1CE2        BNEQ   65$                        ; If so, don't output the time yet.
                017B  30  1CE4        BSBW   OUTPUT_TIME                ; Otherwise, do it now.
         00002248'EF  D6  1CE7  65$:  INCL   LINE_NUM                   ; Update line number.
                  4E  11  1CED        BRB    100$                       ; Branch to common code.
         000021C4'EF  D0  1CEF  70$:  MOVL   HEADER_SIZE,-              ; Save size of SCS header.
         000021CC'EF      1CF5               SCS_HEADER_SIZE            ;
         00000425'EF  90  1CFA        MOVB   NUM_FIELDS,-               ; Save the number of SCS fields being
         00000426'EF      1D00               SCS_NUM_FIELDS             ;  displayed.
                  00  E1  1D05  80$:  BBC    #CLS$V_CLUB,-              ; If CLUB bit is clear we need to output
      24 000021FC'EF      1D07               SAVED_CLASSES,90$          ;  the display descriptor.
         00000426'EF  91  1D0D        CMPB   SCS_NUM_FIELDS,-           ; Are there more SCS fields than CLUB
         00000427'EF      1D13               CLUB_NUM_FIELDS            ;  fields beiJng displayed?
                  0B  18  1D18        BGEQ   85$                        ; Branch if so.
         00000427'EF  90  1D1A        MOVB   CLUB_NUM_FIELDS,-          ; Otherwise, limit left-scrolling by the
         00000425'EF      1D20               NUM_FIELDS                 ;  number of CLUB fields.
                013A  30  1D25  85$:  BSBW   OUTPUT_TIME                ; Output the time.
 00002248'EF      08  C0  1D28        ADDL   #8,LINE_NUM                ; Skip past the CLUB display area.
                  0C  11  1D2F        BRB    100$                       ; Branch to common code.
                0063  30  1D31  90$:  BSBW   OUTPUT_DDM                 ; Output the display descriptor message.
                012B  30  1D34        BSBW   OUTPUT_TIME                ; Output the time.
         00002248'EF  D6  1D37        INCL   LINE_NUM                   ; Update line number.
                          1D3D
   5B    00000763'EF  DE  1D3D 100$:  MOVAL  CLASS_BOX_DSCR,R11         ; Point to descriptor for top of box.
                071F  30  1D44        BSBW   DRAW_LINE                  ; Output top of box.
```

```
         48 50    E9  1D47  5795          BLBC    R0,999$                 ; Branch on error.
 5B  00001593'EF  DE  1D4A  5796          MOVAL   CLASS_DSCR,R11          ; Point to descriptor for the classes.
           0712   30  1D51  5797          BSBW    DRAW_LINE               ; Output class names.
         3B 50    E9  1D54  5798          BLBC    R0,999$                 ; Branch on error.
 5B  0000096B'EF  DE  1D57  5799          MOVAL   FIELD_TOP_BOX_DSCR,R11  ; Point to descriptor for line above
                      1D5E  5800                                          ;  the field names.
           0705   30  1D5E  5801          BSBW    DRAW_LINE               ; Output the line.
         2E 50    E9  1D61  5802          BLBC    R0,999$                 ; Branch on error.
 5B  0000118B'EF  DE  1D64  5803          MOVAL   TITLE_DSCR,R11          ; Point to descriptor for the field
                      1D6B  5804                                          ;  titles.
           06F8   30  1D6B  5805          BSBW    DRAW_LINE               ; Output the field names.
         21 50    E9  1D6E  5806          BLBC    R0,999$                 ; Branch on error.
09 00002247'EF 01 E1  1D71  5807          BBC     #DVT$V_DCRT,OUT_DEV,110$; Branch if terminal is not a DECcrt.
 5B  00000B73'EF  DE  1D79  5808          MOVAL   FIELD_BOT_BOX_DSCR,R11  ; Point to descriptor for line below
                      1D80  5809                                          ;  the field names.
             07   11  1D80  5810          BRB     120$                    ; Jump to common code.
 5B  0000096B'EF  DE  1D82  5811 110$:    MOVAL   FIELD_TOP_BOX_DSCR,R11  ; Point to descriptor for line above
                      1D89  5812                                          ;  the field names.
           06DA   30  1D89  5813 120$:    BSBW    DRAW_LINE               ; Complete the box around the field
         03 50    E9  1D8C  5814          BLBC    R0,999$                 ; Branch on error.
                      1D8F  5815                                          ;  names.
         50  01   D0  1D8F  5816 998$:    MOVL    #SS$_NORMAL,R0
        0800 8F   BA  1D92  5817 999$:    POPR    #^M<R11>
                  05  1D96  5818          RSB
                      1D97  5819
                      1D97  5820
```

```
                               1D97          .SBTTL  -       Output the display description
                               1D97 ;++
                               1D97 ;
                               1D97 ; FUNCTIONAL DESCRIPTION:
                               1D97 ;
                               1D97 ;        This routine is called by GET_HEADER only if we are doing a refresh
                               1D97 ;        pass.  It outputs the description of the display which appears on the
                               1D97 ;        top line of the report.
                               1D97 ;
                               1D97 ; CALLING SEQUENCE:
                               1D97 ;
                               1D97 ;        JSB     OUTPUT_DDM
                               1D97 ;
                               1D97 ;
                               1D97 ; CALLED BY:
                               1D97 ;
                               1D97 ;        GET_HEADER
                               1D97 ;
                               1D97 ; OUTPUTS:
                               1D97 ;
                               1D97 ;        Display discription message is output to the screen buffer.
                               1D97 ;        Length of message is stored in DDM_SIZE.
                               1D97 ;
                               1D97 ; IMPLICIT OUTPUTS:
                               1D97 ;
                               1D97 ;        LINE_NUM is set to 1.
                               1D97 ;
                               1D97 ; RETURN STATUS:
                               1D97 ;
                               1D97 ;        R0 =    SS$_Normal or failed status from LIB$PUT_SCREEN.
                               1D97 ;
                               1D97 ;
                               1D97 OUTPUT_DDM:
              048C 8F    BB    1D97          PUSHR   #^M<R2,R3,R7,R10>
                   53    D4    1D9B          CLRL    R3                      ; Initialize string length.
  00002248'EF      01    D0    1D9D          MOVL    #1,LINE_NUM             ; Output description on line one.
        0000224C'EF      3F    1DA4          PUSHAW  COL_NUM                 ; Setup for LIB$PUT_SCREEN.
        00002248'EF      3F    1DAA          PUSHAW  LINE_NUM                ;
                   00    E1    1DB0          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
     09 00000000'EF            1DB2                  CURRENT_REPORT,10$      ;  cluster report.
  52    0000018A'EF      DE    1DB8          MOVAL   CLUSTER_MSG,R2          ; Get address of message
                               1DBF                                          ;  "View of Cluster from" string.
                   07    11    1DBF          BRB     20$                     ; Jump to common code.
  52    000001A6'EF      DE    1DC1 10$:     MOVAL   LOCAL_MSG,R2            ; Get address of "Local ports on".
             53    62    A0    1DC8 20$:     ADDW    (R2),R3                 ; Calculate string length thus far.
                   52    DD    1DCB          PUSHL   R2                      ; Output the message.
  00000000'GF      03    FB    1DCD          CALLS   #3,G^LIB$PUT_SCREEN     ;
                               1DD4          JLBC    R0,99$                  ; Branch on error.
  53    000001BC'EF      A0    1DDA          ADDW    MSG1_DSCR,R3            ; Add length of system id message to
                               1DE1                                          ;  string length.
        000001BC'EF      DF    1DE1          PUSHAL  MSG1_DSCR               ; Push address of " system ID# "
                               1DE7                                          ;  string.
  00000000'GF      01    FB    1DE7          CALLS   #1,G^LIB$PUT_SCREEN     ; Output the message.
                               1DEE          JLBC    R0,99$                  ; Branch on error.
  5A    00000000'EF      DE    1DF4          MOVAL   LOCSYSID_DSCR,R10       ; Point to descriptor for the local
                               1DFB                                          ;  system ID.
        57    10 AA      D0    1DFB          MOVL    FD$L_OFFSET(R10),R7     ; Get the address of the data.
```

```
          077D  30  1DFF            BSBW    FORMAT_DATA             ; Format the data.
                    1E02            JLBC    R0,99$                  ; Branch on error.
53   00002153'EF  A0  1E08          ADDW    FORMAT_BUF_DSCR,R3      ; Add length of system id to string
                    1E0F                                            ;  length.
     00002153'EF  DF  1E0F          PUSHAL  FORMAT_BUF_DSCR         ; Push the address of the formatted
                    1E15                                            ;  data.
00000000'GF   01  FB  1E15          CALLS   #1,G^LIB$PUT_SCREEN     ; Output the system ID number.
                    1E1C            JLBC    R0,99$                  ; Branch on error.
53   000001CF'EF  A0  1E22          ADDW    MSG2_DSCR,R3            ; Add length of node message to string
                    1E29                                            ;  length.
     000001CF'EF  DF  1E29          PUSHAL  MSG2_DSCR               ; Push the address of "  node: "
                    1E2F                                            ;  string.
00000000'GF   01  FB  1E2F          CALLS   #1,G^LIB$PUT_SCREEN     ; Output the message.
                    1E36            JLBC    R0,99$                  ; Branch on error.
53   00000448'EF  A0  1E3C          ADDW    NODE_NAME_DSCR,R3       ; Add length of node name to string
                    1E43                                            ;  length.
     00000448'EF  DF  1E43          PUSHAL  NODE_NAME_DSCR          ; Push address of local nodename.
00000000'GF   01  FB  1E49          CALLS   #1,G^LIB$PUT_SCREEN     ; Output the nodename.
                    1E50            JLBC    R0,99$                  ; Branch on error.
000020D1'EF   53  B0  1E56          MOVW    R3,DDM_SIZE             ; Save size of display descriptor
                    1E5D                                            ;  message.
          048C 8F  BA  1E5D   99$:  POPR    #^M<R2,R3,R7,R10>
                05  1E61            RSB
                    1E62
                    1E62
```

```
                                  1E62          .SBTTL  -       Output the time
                                  1E62  ;++
                                  1E62  ;
                                  1E62  ; FUNCTIONAL DESCRIPTION:
                                  1E62  ;
                                  1E62  ;       This routine is called by GET_HEADER to output the time on the
                                  1E62  ;       top line of the display.
                                  1E62  ;
                                  1E62  ;       If the size of the either the CLUSTER class header, the SCS class
                                  1E62  ;       header in the CLUSTER report, or the header for the LOCAL report
                                  1E62  ;       (depending on what's being displayed) is larger that the display
                                  1E62  ;       description message + 10, then the time is output along the right
                                  1E62  ;       hand border of the header.  If it's smaller, then the time is
                                  1E62  ;       output at the extreme right.  This is done to improve the appearance
                                  1E62  ;       of the display and to allow hardcopy output on 80 column paper without
                                  1E62  ;       cutting off the time.
                                  1E62  ;
                                  1E62  ;
                                  1E62  ; CALLING SEQUENCE:
                                  1E62  ;
                                  1E62  ;       JSB     OUTPUT_TIME
                                  1E62  ;
                                  1E62  ; CALLED BY:
                                  1E62  ;
                                  1E62  ;       GET_HEADER
                                  1E62  ;
                                  1E62  ; INPUTS:
                                  1E62  ;
                                  1E62  ;       DDM_SIZE contains the length of the display description message.
                                  1E62  ;       SCREEN_WIDTH contains the current width of the screen.
                                  1E62  ;       HEADER_SIZE, CLUB_HEADER_SIZE and SCS_HEADER_SIZE contain the
                                  1E62  ;       width of the header for the classes being displayed.
                                  1E62  ;
                                  1E62  ; OUTPUTS:
                                  1E62  ;
                                  1E62  ;       Time is output to the screen buffer.
                                  1E62  ;
                                  1E62  ; IMPLICIT OUTPUTS:
                                  1E62  ;
                                  1E62  ;       LINE_NUM is set to 1.
                                  1E62  ;
                                  1E62  ; RETURN STATUS:
                                  1E62  ;
                                  1E62  ;       R0 =    SS$_Normal or failed status from LIB$PUT_SCREEN.
                                  1E62  ;
                                  1E62
                                  1E62  OUTPUT_TIME:
                         3C   BB  1E62          PUSHR   #^M<R2,R3,R4,R5>
             00002248'EF 01   D0  1E64          MOVL    #1,LINE_NUM                     ;
0000224C'EF  000020D1'EF 01   A1  1E6B          ADDW3   #1,DDM_SIZE,COL_NUM             ; Get number of column after display
                                  1E77                                                  ;  description message.
             00002248'EF 01   B0  1E77          MOVW    #1,LINE_NUM                     ; Time goes on line one.
                         00   E1  1E7E          BBC     #RPT$V_CLSTR,-                  ; Branch if not displaying the cluster
             1F 00000000'EF       1E80                  CURRENT_REPORT,10$              ;  report.
                000021CC'EF   D1  1E86          CMPL    SCS_HEADER_SIZE,-               ; Use the larger header to determine
                000021C8'EF       1E8C                  CLUB_HEADER_SIZE                ;  where the time should be output.
                         09   14  1E91          BGTR    5$                              ; Branch if SCS is larger.
```

```
                 54   000021C8'EF   D0  1E93          MOVL    CLUB_HEADER_SIZE,R4    ; Get the header size of the CLUB.
                              10    11  1E9A          BRB     20$                    ; Branch to common code.
                 54   000021CC'EF   D0  1E9C  5$:     MOVL    SCS_HEADER_SIZE,R4     ; Get the header size of the SCS.
                              07    11  1EA3          BRB     20$                    ; Branch to common code.
                 54   000021C4'EF   D0  1EA5  10$:    MOVL    HEADER_SIZE,R4         ; Get the header size for the local
                                        1EAC                                         ;  report.
                                        1EAC  ;
                                        1EAC  ; Determine how many spaces to output between the display discription message
                                        1EAC  ; and the time based on the size on the header.  If it's less than 10, then
                                        1EAC  ; output the time at the extreme right.
                                        1EAC  ;
                                        1EAC
52    000020D1'EF   000020B5'EF     A1  1EAC  20$:    ADDW3   TIME_DSCR,DDM_SIZE,R2  ; Get size of message + time.
                  53     54   52    A3  1EB8          SUBW3   R2,R4,R3               ; Get the number of columns to erase.
                         06   53    B1  1EBC          CMPW    R3,#6                  ; Less than 6 blanks?
                              18    18  1EBF          BGEQ    40$                    ; Branch if not.
           0A 00002247'EF     02    E0  1EC1          BBS     #DVT$V_HARD,OUT_DEV,30$ ; Branch if output is hardcopy.
         53   00002222'EF     52    A3  1EC9          SUBW3   R2,SCREEN_WIDTH,R3     ; Get the number of columns to erase.
                              06    11  1ED1          BRB     40$                    ; Branch to common code.
              53   0050 8F    52    A3  1ED3  30$:    SUBW3   R2,#80,R3              ; Get the number of columns to erase.
                              53    D5  1ED9  40$:    TSTL    R3                     ; Skip decrement if fill length is 0.
                              02    13  1EDB          BEQL    50$                    ;
                              53    B7  1EDD          DECW    R3                     ; One less for aesthetics...
              0000199B'EF     53    B0  1EDF  50$:    MOVW    R3,DATA_DSCR           ; Use data descriptor for fill.
              0000199F'FF     00    2C  1EE6          MOVC5   #0,@DATA_DSCR+4,-      ; Fill the descriptor with some spaces.
         0000199F'FF     53   20        1EED                  #^A/ /,R3,@DATA_DSCR+4 ;
              0000224C'EF           3F  1EF4          PUSHAW  COL_NUM                ; Setup for LIB$PUT_SCREEN.
              00002248'EF           3F  1EFA          PUSHAW  LINE_NUM               ;
              0000199B'EF           DF  1F00          PUSHAL  DATA_DSCR              ; Output the spaces.
              00000000'GF     03    FB  1F06          CALLS   #3,G^LIB$PUT_SCREEN    ;
                                        1F0D          JLBC    R0,99$                 ; Branch on error.
              000020B5'EF           DF  1F13          PUSHAL  TIME_DSCR              ; Push address to store time.
              00000000'GF     01    FB  1F19          CALLS   #1,G^LIB$DATE_TIME     ; Get the current date and time.
                         17   50    E9  1F20          BLBC    R0,99$                 ; Branch on error.
              000020B5'EF           DF  1F23          PUSHAL  TIME_DSCR              ;
              00000000'GF     01    FB  1F29          CALLS   #1,G^LIB$PUT_SCREEN    ;
                         07   50    E9  1F30          BLBC    R0,99$                 ; Branch on error.
              00000000'GF     00    FB  1F33          CALLS   #0,G^LIB$ERASE_LINE    ; Erase the rest of the line.
              0000224C'EF     01    D0  1F3A  99$:    MOVL    #1,COL_NUM             ; Restore column number.
                              3C    BA  1F41          POPR    #^M<R2,R3,R4,R5>
                                    05  1F43          RSB
                                        1F44
                                        1F44
```

```
                                  1F44          .SBTTL          -       Box drawing routines.
                                  1F44   ;++
                                  1F44   ;
                                  1F44   ; FUNCTIONAL DESCRIPTION:
                                  1F44   ;
                                  1F44   ;       This routine builds the descriptors for the lines which make up the
                                  1F44   ;       boxes surrounding the classes and titles in the display header and the
                                  1F44   ;       line following the last line of data in the report when the report is
                                  1F44   ;       run on a non-VT100 compatable terminal.
                                  1F44   ;
                                  1F44   ; CALLING SEQUENCE:
                                  1F44   ;
                                  1F44   ;       JSB     BUILD_VT52_BOX
                                  1F44   ;
                                  1F44   ; CALLED BY:
                                  1F44   ;
                                  1F44   ;       GET_HEADER
                                  1F44   ;
                                  1F44   ; OUTPUTS:
                                  1F44   ;       Data for line above and below the titles and the line following the
                                  1F44   ;       last line of data stored in FIELD_TOP_BOX_BUF_DSCR.
                                  1F44   ;       Data for line above and below the classes stored in CLASS_BOX_BUF_DSCR.
                                  1F44   ;
                                  1F44   ; RETURN STATUS:
                                  1F44   ;
                                  1F44   ;       None
                                  1F44   ;
                                  1F44   ; REGISTER USAGE:
                                  1F44   ;
                                  1F44   ;       R5 -    Pointer into TITLE_BUF
                                  1F44   ;       R6 -    Counter for the number of charcters in the buffers.
                                  1F44   ;       R7 -    Pointer into CLASS_BUF
                                  1F44   ;       R8 -    Pointer into CLASS_BOX_BUF
                                  1F44   ;       R9 -    Pointer into FIELD_TOP_BOX_BUF
                                  1F44   ;
                                  1F44   ;--
                                  1F44
                                  1F44   BUILD_VT52_BOX:
         03E0 8F     BB           1F44          PUSHR   #^M<R5,R6,R7,R8,R9>
58   0000076B'EF     DE           1F48          MOVAL   CLASS_BOX_BUF,R8                ; Get address of buffer for line above
                                  1F4F                                                  ;  the classes.
59   00000973'EF     DE           1F4F          MOVAL   FIELD_TOP_BOX_BUF,R9            ; Get address of buffer for above and
                                  1F56                                                  ;  below field titles and last line.
56   0000118B'EF     3C           1F56          MOVZWL  TITLE_DSCR,R6                   ; Get the number of bytes in TITLE_BUF.
0000096B'EF    56    B0           1F5D          MOVW    R6,FIELD_TOP_BOX_DSCR           ; Save as the number of bytes in
                                  1F64                                                  ;  FIELD_BOX_BUF.
00000763'EF    56    B0           1F64          MOVW    R6,CLASS_BOX_DSCR               ; Save as the number of bytes in
                                  1F6B                                                  ;  CLASS_BOX_BUF.
57   0000159B'EF     DE           1F6B          MOVAL   CLASS_BUF,R7                    ; Get the address of the class buffer.
55   00001193'EF     DE           1F72          MOVAL   TITLE_BUF,R5                    ; Get the address of the title buffer.
      7C 8F    87    91           1F79   10$:   CMPB    (R7)+,#^A/!/                    ; Is this character a vertical bar?
               05    12           1F7D          BNEQU   20$                             ; Branch if not.
         88    2B    90           1F7F          MOVB    #^A/+/,(R8)+                    ; Store a plus sign in the buffer.
               03    11           1F82          BRB     30$
         88    2D    90           1F84   20$:   MOVB    #^A/-/,(R8)+                    ; Store a minus sign in the buffer.
      7C 8F    85    91           1F87   30$:   CMPB    (R5)+,#^A/!/                    ; Is this character a vertical bar?
               05    12           1F8B          BNEQU   40$                             ; Branch if not.
```

```
89      2B   90  1F8D              MOVB    #^A/+/,(R9)+              ; Store a plus sign in the buffer.
        03   11  1F90              BRB     50$
89      2D   90  1F92       40$:   MOVB    #^A/-/,(R9)+              ; Store a minus sign in the buffer.
   E1   56   F5  1F95       50$:   SOBGTR  R6,10$                    ; Loop for all characters.
03E0    8F   BA  1F98              POPR    #^M<R5,R6,R7,R8,R9>
             05  1F9C              RSB
                 1F9D
                 1F9D
```

```
                          1F9D  ;++
                          1F9D  ;
                          1F9D  ; FUNCTIONAL DESCRIPTION:
                          1F9D  ;
                          1F9D  ;       This routine builds the descriptors for the lines which make up the
                          1F9D  ;       boxes surrounding the classes and titles in the display header and the
                          1F9D  ;       line following the last line of data in the report when the report is
                          1F9D  ;       run on a VT100 compatable terminal.
                          1F9D  ;
                          1F9D  ; CALLING SEQUENCE:
                          1F9D  ;
                          1F9D  ;       JSB     BUILD_DECCRT_BOX
                          1F9D  ;
                          1F9D  ; CALLED BY:
                          1F9D  ;
                          1F9D  ;       GET_HEADER
                          1F9D  ;
                          1F9D  ; OUTPUTS:
                          1F9D  ;
                          1F9D  ;       Date for line above classes stored in CLASS_BOX_BUF.
                          1F9D  ;       Data for line above fields stored in FIELD_TOP_BOX_BUF.
                          1F9D  ;       Data for line below fields stored in FIELD_BOT_BOX_BUF.
                          1F9D  ;       Data for line below the last line of data stored in
                          1F9D  ;       LAST_LINE_BUF for SCS display or LAST_CLUB_LINE_BUF for
                          1F9D  ;       CLUB display.
                          1F9D  ;
                          1F9D  ; RETURN STATUS:
                          1F9D  ;
                          1F9D  ;       None.
                          1F9D  ;
                          1F9D  ; REGISTER USAGE:
                          1F9D  ;
                          1F9D  ;       R2 -    Pointer into CLASS_BOX_BUF.
                          1F9D  ;       R3 -    Pointer into FIELD_TOP_BOX_BUF.
                          1F9D  ;       R4 -    Pointer into FIELD_BOT_BOX_BUF.
                          1F9D  ;       R5 -    Pointer into LAST_LINE_BUF or LAST_CLUB_LINE_BOX.
                          1F9D  ;       R6 -    Counter for the number of characters in the buffers.
                          1F9D  ;       R7 -    Pointer into CLASS_BUF.
                          1F9D  ;       R8 -    Pointer into TITLE_BUF.
                          1F9D  ;--
                          1F9D
                          1F9D  BUILD_DECCRT_BOX:
        01FC 8F     BB    1F9D          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>
56  0000118B'EF     3C    1FA1          MOVZWL  TITLE_DSCR,R6           ; Get the number of bytes in TITLE_BUF.
                          1FA8          JEQL    60$                     ; Branch if nothing to display.
52  0000076B'EF     DE    1FAD          MOVAL   CLASS_BOX_BUF,R2        ; Get address of buffer for line above
                          1FB4                                          ;  classes.
53  00000973'EF     DE    1FB4          MOVAL   FIELD_TOP_BOX_BUF,R3    ; Get address of buffer for line above
                          1FBB                                          ;  fields.
54  00000B7B'EF     DE    1FBB          MOVAL   FIELD_BOT_BOX_BUF,R4    ; Get address of buffer for line below
                          1FC2                                          ;  fields.
            00      E1    1FC2          BBC     #CLS$V_CLUB,-           ; Branch if not displaying the CLUSTER
   09 00000001'EF         1FC4                  CLASS_SELECT,5$         ;  class.
55  00000F8B'EF     DE    1FCA          MOVAL   LAST_CLUB_LINE_BUF,R5   ; Get address of buffer for line below
                          1FD1                                          ;  the last data line of CLUB display.
            07      11    1FD1          BRB     6$                      ; Branch to common code.
55  00000D83'EF     DE    1FD3  5$:     MOVAL   LAST_LINE_BUF,R5        ; Get address of buffer for line below
```

```
                           1FDA  6129                                               ;  the last data line of SCS display.
       57 0000159B'EF  DE  1FDA  6130 6$:     MOVAL   CLASS_BUF,R7                  ; Get address of buffer for class
                           1FE1  6131                                               ;  names.
       58 00001193'EF  DE  1FE1  6132         MOVAL   TITLE_BUF,R8                  ; Get address of buffer for field
                           1FE8  6133                                               ;  names.
       56 0000118B'EF  3C  1FE8  6134         MOVZWL  TITLE_DSCR,R6                 ; Get the number of bytes in TITLE_BUF.
             82    0E  90  1FEF  6135         MOVB    #GRAPHICS_ON,(R2)+            ; Store character to turn on graphics in
             83    0E  90  1FF2  6136         MOVB    #GRAPHICS_ON,(R3)+            ;  each of the buffers.
             84    0E  90  1FF5  6137         MOVB    #GRAPHICS_ON,(R4)+            ;
             85    0E  90  1FF8  6138         MOVB    #GRAPHICS_ON,(R5)+            ;
                   01  E0  1FFB  6139         BBS     #SCR$V_MORE_LFT,-             ; Branch if screen has been scrolled
       19 0000047A'EF      1FFD  6140                 SCROLL_FLAGS,10$              ;  left.
          82    6C 8F  90  2003  6141         MOVB    #UL_CORNER,(R2)+              ; Store upper-left corner.
          83    74 8F  90  2007  6142         MOVB    #LEFT_T,(R3)+                 ; Store left "T".
          84    74 8F  90  200B  6143         MOVB    #LEFT_T,(R4)+                 ; Store left "T".
          85    6D 8F  90  200F  6144         MOVB    #LL_CORNER,(R5)+              ; Store lower-left corner.
             57    03  C0  2013  6145         ADDL    #DELM_DSCR_LEN,R7             ; Ignore first character in CLASS_BUF
             58    03  C0  2016  6146         ADDL    #DELM_DSCR_LEN,R8             ;  and FIELD_BUF. (assumed a "|").
             56    03  C2  2019  6147         SUBL    #DELM_DSCR_LEN,R6             ; Decrement number of bytes remaining.
             88    0E  91  201C  6148 10$:    CMPB    #GRAPHICS_ON,(R8)+            ; Is there a field delimiter in
                           201F  6149                                               ;  FIELD_BUF?
                   2A  12  201F  6150         BNEQU   30$                           ; Branch if not.
             58    02  C0  2021  6151         ADDL    #DELM_DSCR_LEN-1,R8           ; Update pointer into FIELD_BUF.
             56    02  C2  2024  6152         SUBL    #DELM_DSCR_LEN-1,R6           ; Update counter.
          84    6E 8F  90  2027  6153         MOVB    #CROSS,(R4)+                  ; Store crossing lines in
                           202B  6154                                               ;  FIELD_BOT_BOX_BUF.
          85    76 8F  90  202B  6155         MOVB    #BOTTOM_T,(R5)+               ; Store a bottom "T" in LAST_LINE_BUF.
             87    0E  91  202F  6156         CMPB    #GRAPHICS_ON,(R7)+            ; Is there an escape sequence in
                           2032  6157                                               ;  CLASS_BUF?
                   0D  12  2032  6158         BNEQU   20$                           ; Branch if not.
          83    6E 8F  90  2034  6159         MOVB    #CROSS,(R3)+                  ; Store crossing lines in
                           2038  6160                                               ;  FIELD_TOP_BOX_BUF.
          82    77 8F  90  2038  6161         MOVB    #TOP_T,(R2)+                  ; Store a top "T" in CLASS_BOX_BUF.
             57    02  C0  203C  6162         ADDL    #DELM_DSCR_LEN-1,R7           ; Update pointer in CLASS_BUF.
                   1C  11  203F  6163         BRB     40$                           ; Join common code.
                           2041  6164
          83    77 8F  90  2041  6165 20$:    MOVB    #TOP_T,(R3)+                  ; Store a top "T" in FIELD_TOP_BOX_BUF.
          82    71 8F  90  2045  6166         MOVB    #SCAN_5,(R2)+                 ; Store a horizontal line - scan 5 in
                           2049  6167                                               ;  CLASS_BOX_BUF.
                   12  11  2049  6168         BRB     40$                           ; Join common code.
                           204B  6169
          82    71 8F  90  204B  6170 30$:    MOVB    #SCAN_5,(R2)+                 ; Store a horizontal line - scan 5 in
                           204F  6171                                               ;  CLASS_BOX_BUF...
          83    71 8F  90  204F  6172         MOVB    #SCAN_5,(R3)+                 ;   in FIELD_TOP_BOX_BUF...
          84    71 8F  90  2053  6173         MOVB    #SCAN_5,(R4)+                 ;   in FIELD_BOT_BOX_BUF...
          85    71 8F  90  2057  6174         MOVB    #SCAN_5,(R5)+                 ;   and in LAST_LINE_BUF.
                   57  D6  205B  6175         INCL    R7                            ; Update pointer into CLASS_BUF.
             BC    56  F5  205D  6176 40$:    SOBGTR  R6,10$                        ; Loop for all bytes in CLASS_BUF.
                   52  D7  2060  6177         DECL    R2                            ; Now that we know we've hit the end
                   53  D7  2062  6178         DECL    R3                            ;  backup one character in each buffer
                   54  D7  2064  6179         DECL    R4                            ;  and fix the last character on the
                   55  D7  2066  6180         DECL    R5                            ;  line.
          0F6B 8F  B0  2068  6181         MOVW    #<GRAPHICS_OFF@8>!UR_CORNER,-
                   82      206C  6182                 (R2)+                         ; Store an upper-right corner followed
                           206D  6183                                               ;   by character to turn graphics off in
                           206D  6184                                               ;  CLASS_BOX_BUF.
          0F75 8F  B0  206D  6185         MOVW    #<GRAPHICS_OFF@8>!RIGHT_T,-
```

```
                    83           2071  6186              (R3)+                      ; Store a right "T" followed by
                                 2072  6187                                         ;  character to turn graphics off
                                 2072  6188                                         ;  in FIELD_TOP_BOX_BUF.
               0F75 8F    B0     2072  6189       MOVW   #<GRAPHICS_OFF@8>!RIGHT_T,-
                    84           2076  6190              (R4)+                      ; Store a right "T" followed by
                                 2077  6191                                         ;  character to turn graphics off in
                                 2077  6192                                         ;  FIELD_BOT_BOX_BUF.
               0F6A 8F    B0     2077  6193       MOVW   #<GRAPHICS_OFF@8>!LR_CORNER,-
                    85           207B  6194              (R5)+                      ; Store a lower-right corner followed
                                 207C  6195                                         ;  by character to turn graphics off in
                                 207C  6196                                         ;  LAST_LINE_BUF.
                                 207C  6197
00000763'EF   52  00000767'EF C3 207C  6198       SUBL3  CLASS_BOX_DSCR+4,R2,-      ; Save the number of characters stored
                                 2088  6199              CLASS_BOX_DSCR             ;  in CLASS_BOX_BUF.
              0000096F'EF C3     2088  6200       SUBL3  FIELD_TOP_BOX_DSCR+4,-     ; Save the number of characters stored
         0000096B'EF  53         208E  6201              R3,FIELD_TOP_BOX_DSCR      ;  in FIELD_TOP_BOX_BUF.
              00000B77'EF C3     2094  6202       SUBL3  FIELD_BOT_BOX_DSCR+4,-     ; Save the number of characters stored
         00000B73'EF  54         209A  6203              R4,FIELD_BOT_BOX_DSCR      ;  in FIELD_BOT_BOX_BUF.
                       00 E1     20A0  6204       BBC    #CLS$V_CLUB,-              ; Branch if not displaying the CLUSTER
              0E 00000001'EF     20A2  6205              CLASS_SELECT,50$           ;  class.
              00000F87'EF C3     20A8  6206       SUBL3  LAST_CLUB_LINE_DSCR+4,-    ; Save the number of characters stored
         00000F83'EF  55         20AE  6207              R5,LAST_CLUB_LINE_DSCR     ;  in LAST_CLUB_LINE_BUF.
                    36    11     20B4  6208       BRB    70$                        ; Branch to common code.
00000D7B'EF   55  00000D7F'EF C3 20B6  6209 50$:  SUBL3  LAST_LINE_DSCR+4,R5,-      ; Save the number of characters stored
                                 20C2  6210              LAST_LINE_DSCR             ;  in LAST_LINE_BUF.
                    28    11     20C2  6211       BRB    70$                        ; Branch to common code.
                                 20C4  6212
              00000763'EF D4     20C4  6213 60$:  CLRL   CLASS_BOX_DSCR             ; Nothing to display, so clear length in
              0000096B'EF D4     20CA  6214       CLRL   FIELD_TOP_BOX_DSCR         ;  all descriptors.
              00000B73'EF D4     20D0  6215       CLRL   FIELD_BOT_BOX_DSCR         ;
                       00 E1     20D6  6216       BBC    #CLS$V_CLUB,-              ; Branch if not displaying the CLUSTER
              08 00000001'EF     20D8  6217              CLASS_SELECT,65$           ;  class.
              00000F83'EF D4     20DE  6218       CLRL   LAST_CLUB_LINE_DSCR        ;
                       06 11     20E4  6219       BRB    70$                        ; Branch to common code.
              00000D7B'EF D4     20E6  6220 65$:  CLRL   LAST_LINE_DSCR             ;
               01FC 8F    BA     20EC  6221 70$:  POPR   #^M<R2,R3,R4,R5,R6,R7,R8>
                       05        20F0  6222       RSB
                                 20F1  6223
                                 20F1  6224
```

```
                                 20F1  6226          .SBTTL          -       DRAW_LAST_LINE
                                 20F1  6227
                                 20F1  6228 ;++
                                 20F1  6229 ;
                                 20F1  6230 ; FUNCTIONAL DESCRIPTION:
                                 20F1  6231 ;
                                 20F1  6232 ;       This routine is used to output the line immediately following the
                                 20F1  6233 ;       data, i.e., the bottom line of the display box.
                                 20F1  6234 ;
                                 20F1  6235 ; CALLING SEQUENCE:
                                 20F1  6236 ;
                                 20F1  6237 ;       JSB     DRAW_LAST_LINE
                                 20F1  6238 ;
                                 20F1  6239 ; CALLED BY:
                                 20F1  6240 ;
                                 20F1  6241 ;       DISPLAY_CLUB
                                 20F1  6242 ;       DISPLAY_COMMON
                                 20F1  6243 ;       GET_HELP
                                 20F1  6244 ;
                                 20F1  6245 ; INPUTS:
                                 20F1  6246 ;
                                 20F1  6247 ;       LAST_LINE_DSCR or LAST_CLUB_LINE_DSCR filled in for DECcrts.
                                 20F1  6248 ;       FIELD_TOP_BOX_DSCR filled in if not DECcrt.
                                 20F1  6249 ;
                                 20F1  6250 ; OUTPUTS:
                                 20F1  6251 ;
                                 20F1  6252 ;       One line output to SCREEN_BUFFER.
                                 20F1  6253 ;
                                 20F1  6254 ; RETURN STATUS:
                                 20F1  6255 ;
                                 20F1  6256 ;       R0 = Status returned form DRAW_LINE routine.
                                 20F1  6257 ;
                                 20F1  6258 ; REGISTER USAGE:
                                 20F1  6259 ;
                                 20F1  6260 ;       R11 points to descriptor for the line to be output.
                                 20F1  6261 ;
                                 20F1  6262 ;--
                                 20F1  6263
                                 20F1  6264 DRAW_LAST_LINE:
              0800 8F      BB    20F1  6265         PUSHR   #^M<R11>
   38 00002247'EF    01    E1    20F5  6266         BBC     #DVT$V_DCRT,OUT_DEV,30$ ; Branch if output device isn't a
                                 20FD  6267                                         ;  DECcrt terminal.
                    00     E1    20FD  6268         BBC     #RPT$V_CLSTR,-          ; Branch if not displaying the Cluster
       27 00000000'EF            20FF  6269                 CURRENT_REPORT,20$      ;  report.
                    00     E1    2105  6270         BBC     #CLS$V_CLUB,-           ; Branch if not displaying the CLUSTER
       14 00000001'EF            2107  6271                 CLASS_SELECT,10$        ;  class.
   5B    00000F83'EF       DE    210D  6272         MOVAL   LAST_CLUB_LINE_DSCR,R11 ; Point to descriptor for last line
                                 2114  6273                                         ;  of CLUB display.
         000021C8'EF       D0    2114  6274         MOVL    CLUB_HEADER_SIZE,-      ; Get size of CLUB header.
         000021C4'EF             211A  6275                 HEADER_SIZE
                 1B        11    211F  6276         BRB     40$                     ; Branch to common code.
         000021CC'EF       D0    2121  6277 10$:    MOVL    SCS_HEADER_SIZE,-       ; Get size of SCS header.
         000021C4'EF             2127  6278                 HEADER_SIZE             ;
   5B    00000D7B'EF       DE    212C  6279 20$:    MOVAL   LAST_LINE_DSCR,R11      ; Point to descriptor for last line
                                 2133  6280                                         ;  of SCS display.
                 07        11    2133  6281         BRB     40$                     ; Jump to common code.
   5B    0000096B'EF       DE    2135  6282 30$:    MOVAL   FIELD_TOP_BOX_DSCR,R11  ; Point to descriptor for line below
```

```
                 213C  6283                                   ;  the field names.
      0327  30   213C  6284 40$:    BSBW    DRAW_LINE         ; Output the last line.
 0800 8F    BA   213F  6285         POPR    #^M<R11>
            05   2143  6286         RSB
                 2144  6287
                 2144  6288
```

```
                              2144          .SBTTL          -       Field title generation.
                              2144  ;++
                              2144  ;
                              2144  ; FUNCTIONAL DESCRIPTION:
                              2144  ;
                              2144  ;       This routine scans the field descriptor bit map to determine which
                              2144  ;       fields will be displayed.  If a field is to be displayed, then its
                              2144  ;       title is added to the title buffer along with the appropriate
                              2144  ;       delimiters and spaces so that the title is centered in the field.
                              2144  ;       The scan continues until the entire bit map is checked, or the title
                              2144  ;       buffer is full.
                              2144  ;
                              2144  ; CALLING SEQUENCE:
                              2144  ;
                              2144  ;       JSB     GET_TITLES
                              2144  ;
                              2144  ; CALLED BY:
                              2144  ;
                              2144  ;       GET_HEADER
                              2144  ;
                              2144  ; OUTPUTS:
                              2144  ;
                              2144  ;       Field titles are output to TITLE_BUF.  Descriptor for the title is
                              2144  ;       built at TITLE_DSCR.
                              2144  ;
                              2144  ; RETURN STATUS:
                              2144  ;
                              2144  ;       R0 = SS$_NORNAL
                              2144  ;
                              2144  ; REGISTER USAGE:
                              2144  ;
                              2144  ;       R2 -    Used to determine when the located title is in a different
                              2144  ;               class than the previously located title.
                              2144  ;       R3 -    Points to the next free byte in the title buffer.
                              2144  ;       R6 -    Number of character positions left on the line before the
                              2144  ;               screen line overflows.
                              2144  ;       R7 -    Used to hold the size of the current display field.
                              2144  ;       R9 -    Used to calculate the number of spaces needed to center the
                              2144  ;               title in the field.
                              2144  ;       R10 -   Points to the start of the current field descriptor entry.
                              2144  ;               (From FIND_ENTRY routine)
                              2144  ;
                              2144  ;
                              2144  ;--
                              2144
                              2144  GET_TITLES:
           06CC 8F      BB    2144          PUSHR   #^M<R2,R3,R6,R7,R9,R10>
      00000425'EF       94    2148          CLRB    NUM_FIELDS              ; Initialize number of fields found.
             00         E1    214E          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
   0D 00000000'EF             2150                  CURRENT_REPORT,3$       ;  cluster report.
      00000464'EF       D0    2156          MOVL    CLSTR_H_SCROLL,-        ; Get a copy of the scroll value for the
      0000046C'EF             215C                  H_SCROLL_LINE           ;  cluster report.
             08         11    2161          BRB     4$                      ; Branch to common code.
      00000468'EF       D0    2163  3$:     MOVL    LOCAL_H_SCROLL,-        ; Get a copy of the scroll value for the
      0000046C'EF             2169                  H_SCROLL_LINE           ;  local report.
 53   000021D0'EF       DE    216E  4$:     MOVAL   TITLE_LEN_ARRAY,R3      ; Set length of titles to 0.
             34         BB    2175          PUSHR   #^M<R2,R4,R5>           ;
   00    63  00         2C    2177          MOVC5   #0,(R3),#0,-            ;
```

```
          63  24      217B  6347                #4*NUM_CLASSES,(R3)          ;
              34  BA  217D  6348          POPR  #^M<R2,R4,R5>                ;
     0000118B'EF  D4  217F  6349          CLRL  TITLE_DSCR                   ; Set length of buffer to 0.
     000021C4'EF  D4  2185  6350          CLRL  HEADER_SIZE                  ; Set width of header to 0.
  53 00001193'EF  DE  218B  6351          MOVAL TITLE_BUF,R3                 ; Get pointer to title buffer.
 000021C0'EF  53  D0  2192  6352          MOVL  R3,TITLE_START               ; Save address of start of titles.
  56 00002222'EF  3C  2199  6353          MOVZWL SCREEN_WIDTH,R6             ; Get screen width.
              01  E0  21A0  6354          BBS   #SCR$V_MORE_LFT,-            ; Branch if screen has been scrolled
  09 0000047A'EF      21A2  6355                SCROLL_FLAGS,5$              ;  left.
            0274  30  21A8  6356          BSBW  DELM                         ; Start line with field delimiter.
     000021C4'EF  D6  21AB  6357          INCL  HEADER_SIZE                  ; Header size = 1.
 000021BC'EF  56  D0  21B1  6358 5$:      MOVL  R6,TITLE_COUNT               ; Save the number of bytes remaining on
                      21B8  6359                                             ;  the line.
     00002270'EF  D4  21B8  6360          CLRL  START_BIT                    ; Start scanning the field descriptor
                      21BE  6361                                             ;  bitmap at bit zero.
              00  E1  21BE  6362          BBC   #RPT$V_CLSTR,-               ; Branch if we're not displaying the
  09 00000000'EF      21C0  6363                CURRENT_REPORT,7$            ;  cluster report.
 0000226C'EF  02  D0  21C6  6364          MOVL  #CLUSTER_MAP_LEN,MAP_LEN;    ; Save length of cluster bitmap.
              07  11  21CD  6365          BRB   10$                          ; Jump to common code.
 0000226C'EF  00  D0  21CF  6366 7$:      MOVL  #LOCAL_MAP_LEN,MAP_LEN       ; Save length of local bitmap.
            018A  30  21D6  6367 10$:     BSBW  FIND_ENTRY                   ; Locate a field to be displayed.
           6C 50  E9  21D9  6368          BLBC  R0,80$                       ; Branch if field wasn't found.
     00000425'EF  96  21DC  6369          INCB  NUM_FIELDS                   ; Increment number of fields found.
           0C AA  CB  21E2  6370 20$:     BICL3 FD$L_CLASS(R10),-            ; Is this field in a different class?
  52 000021F4'EF      21E5  6371                CURRENT_CLASS,R2             ;
              02  13  21EB  6372          BEQL  30$                          ; Branch if not.
              63  10  21ED  6373          BSB   SAVE_TITLE_LEN               ; Save length of the class title.
           0C AA  D3  21EF  6374 30$:     BITL  FD$L_CLASS(R10),-            ; Is this class enabled?
     00000001'EF      21F2  6375                CLASS_SELECT                 ;
              DD  13  21F7  6376          BEQLU 10$                          ; Branch if not.
            0247  30  21F9  6377          BSBW  ADD_ONE_SPACE                ; Preceed title with atleast one space.
              48  1F  21FC  6378          BLSSU 70$                          ; Branch if it won't fit.
           18 AA  83  21FE  6379          SUBB3 FD$B_TITLE_SIZE(R10),-       ; Get the difference between the number
          59  6A      2201  6380                FD$B_FIELD_SIZE(R10),R9      ; of characters in the title and the
                      2203  6381                                             ;  size of the field.
              02  18  2203  6382          BGEQ  40$                          ; Branch if field size is larger than
                      2205  6383                                             ;  the title.
              59  94  2205  6384          CLRB  R9                           ; Number of spaces needed before and
                      2207  6385                                             ;  after the title equals zero.
          59  59  9A  2207  6386 40$:     MOVZBL R9,R9                       ; Make R9 a longword.
  59   59  FF 8F  9C  220A  6387          ROTL  #-1,R9,R9                    ; Calculate the number of spaces to
                      220F  6388                                             ;  center the title.
            0237  30  220F  6389          BSBW  ADD_SPACES                   ; Add the spaces to the buffer.
              32  1F  2212  6390          BLSSU 70$                          ; Branch if the spaces won't fit on the
                      2214  6391                                             ;  screen.
       03 59  1F  E1  2214  6392          BBC   #31,R9,50$                   ; Branch if even number of spaces.
            0228  30  2218  6393          BSBW  ADD_ONE_SPACE                ; Otherwise, add one more.
              29  1F  221B  6394 50$:     BLSSU 70$                          ; Branch if the won't fit.
       57  18 AA  9A  221D  6395          MOVZBL FD$B_TITLE_SIZE(R10),R7     ; Assume the full title will fit in the
                      2221  6396                                             ;  field.
          6A  57  91  2221  6397          CMPB  R7,FD$B_FIELD_SIZE(R10)      ; Will it really fit?
              03  15  2224  6398          BLEQ  60$                          ; Branch if it will
          57  6A  9A  2226  6399          MOVZBL FD$B_FIELD_SIZE(R10),R7     ; Get the size of the field.
              34  BB  2229  6400 60$:     PUSHR #^M<R2,R4,R5>                ;
  63   19 AA  57  28  222B  6401          MOVC3 R7,FD$T_TITLE(R10),(R3)      ; Copy title to title buffer.
              34  BA  2230  6402          POPR  #^M<R2,R4,R5>                ;
          56  57  82  2232  6403          SUBB  R7,R6                        ; Calculate room left on screen.
```

```
      0F  1F  2235  6404          BLSSU   70$                   ; Branch if it didn't fit.
    020F  30  2237  6405          BSBW    ADD_SPACES            ; Add spaces following the title.
      0A  1F  223A  6406          BLSSU   70$                   ; Branch if the spaces won't fit on the
              223C  6407                                        ;  screen.
    0204  30  223C  6408          BSBW    ADD_ONE_SPACE         ; Follow title with one space.
      05  1F  223F  6409          BLSSU   70$                   ; Branch if it didn't fit.
    01DB  30  2241  6410          BSBW    DELM                  ; Add ending delimiter to line.
      90  1E  2244  6411          BGEQU   10$                   ; Loop if there's more room on the
              2246  6412                                        ;  line.
      56  D4  2246  6413 70$:     CLRL    R6                    ; Number of columns remaining equals
              2248  6414                                        ;  zero.
      08  10  2248  6415 80$:     BSB     SAVE_TITLE_LEN        ; Save the length of the titles.
  50  01  D0  224A  6416          MOVL    #SS$_NORMAL,R0        ; Indicate success.
 06CC 8F  BA  224D  6417          POPR    #^M<R2,R3,R6,R7,R9,R10>
      05      2251  6418          RSB                           ; Return.
              2252  6419
              2252  6420
```

```
2252  ;++
2252  ;
2252  ; FUNCTIONAL DESCRIPTION:
2252  ;
2252  ;       This routine is used to save the total number of columns used by all the
2252  ;       titles in a particular class that are to be displayed.  This number is
2252  ;       used by the GET_CLASSES routine to center the classes.  This routine
2252  ;       also saves the total number of columns that will be used by the report.
2252  ;       Note that this routine is called whenever a title to be displayed has
2252  ;       been found that is in a different class than the previously found title
2252  ;       or when the last title to be displayed has been found.
2252  ;
2252  ; CALLING SEQUENCE:
2252  ;
2252  ;       JSB     SAVE_TITLE_LEN
2252  ;
2252  ; CALLED BY:
2252  ;
2252  ;       GET_TITLES
2252  ;
2252  ; INPUTS:
2252  ;
2252  ;       R3 points to the address following the last byte stored in the title
2252  ;       buffer.
2252  ;       R6 contains the number of characters that can be added to the title
2252  ;       buffer before the number of characters in the buffer exceeds the width
2252  ;       of the screen.
2252  ;
2252  ; IMPLICIT INPUTS:
2252  ;
2252  ;       TITLE_COUNT = Number of characters that can be added to the title
2252  ;               as saved from a previous call to this rouine.  TITLE_COUNT
2252  ;               is initially set to the screen width by GET_TITLES.
2252  ;       TITLE_START = Address in the title buffer where the last group of
2252  ;               titles begins as saved from a previous call to this routine.
2252  ;               TITLE_START initially points to the start of the title buffer.
2252  ;
2252  ; OUTPUTS:
2252  ;
2252  ;       HEADER_SIZE updated to reflect the width of the report in characters.
2252  ;
2252  ; IMPLICIT OUTPUTS:
2252  ;
2252  ;       TITLE_COUNT updated.
2252  ;       TITLE_START updated.
2252  ;
2252  ; RETURN STATUS:
2252  ;
2252  ;       None.
2252  ;
2252  ; REGISTER USAGE:
2252  ;
2252  ;       R2 -    Used to index into an array containing the lengths of the
2252  ;               titles in each of the classes.
2252  ;       R4 -    Used to point to the base address of the array.
2252  ;       R3,R5 - Used to calculate the number of bytes stored in the title
2252  ;               buffer.
```

```
                                2252  6479 ;
                                2252  6480 ;--
                                2252  6481
                                2252  6482 SAVE_TITLE_LEN:
                       34   BB  2252  6483          PUSHR   #^M<R2,R4,R5>
               09      00   EA  2254  6484          FFS     #0,#NUM_CLASSES,-           ; Convert class bit to index value.
      52  000021F4'EF               2257  6485                  CURRENT_CLASS,R2            ;
      54  000021D0'EF          DE  225D  6486          MOVAL   TITLE_LEN_ARRAY,R4          ; Get address of title length array.
6442  000021BC'EF      56   C3  2264  6487          SUBL3   R6,TITLE_COUNT,(R4)[R2]     ; Save length of titles in characters.
   000021C4'EF       6442   C0  226D  6488          ADDL    (R4)[R2],HEADER_SIZE        ; Add to total size of header in
                                2275  6489                                              ;  characters
  55   53  000021C0'EF      C3  2275  6490          SUBL3   TITLE_START,R3,R5           ; Get actual length of titles in bytes.
      0000118B'EF      55   C0  227D  6491          ADDL    R5,TITLE_DSCR               ; Add length to total length of titles.
                  16   50   E9  2284  6492          BLBC    R0,10$                      ; Skip following if FIND_ENTRY failed.
      000021BC'EF      56   D0  2287  6493          MOVL    R6,TITLE_COUNT              ; Save number of bytes remaining.
      000021C0'EF      53   D0  228E  6494          MOVL    R3,TITLE_START              ; Save starting address for next set of
                                2295  6495                                              ;  titles.
               0C      AA   D0  2295  6496          MOVL    FD$L_CLASS(R10),-           ; Set new class to current class.
          000021F4'EF           2298  6497                  CURRENT_CLASS               ;
                       34   BA  229D  6498 10$:     POPR    #^M<R2,R4,R5>
                            05  229F  6499          RSB
                                22A0  6500
                                22A0  6501
```

```
                          22A0          .SBTTL          -       Class title generation.
                          22A0  ;++
                          22A0  ;
                          22A0  ; FUNCTIONAL DESCRIPTION:
                          22A0  ;
                          22A0  ;       This routine stores the class titles in CLASS_BUF for the GET_HEADER
                          22A0  ;       routine.  The class titles are stored with the appropriate field
                          22A0  ;       delimiters and spacing such that the class titles will be displayed
                          22A0  ;       centered above the field titles.
                          22A0  ;
                          22A0  ;
                          22A0  ; CALLING SEQUENCE:
                          22A0  ;
                          22A0  ;       JSB     GET_CLASSES
                          22A0  ;
                          22A0  ; CALLED BY:
                          22A0  ;
                          22A0  ;       GET_HEADER
                          22A0  ;
                          22A0  ; OUTPUTS:
                          22A0  ;
                          22A0  ;       Clases are stored in CLASS_BUF.
                          22A0  ;
                          22A0  ; RETURN STATUS:
                          22A0  ;
                          22A0  ;       R0 =    SS$_Normal.
                          22A0  ;
                          22A0  ;
                          22A0  ; REGISTER USAGE:
                          22A0  ;
                          22A0  ;       R3 -    Points to next free character position in CLASS_BUF.
                          22A0  ;       R4 -    Used to calculate the number of leading spaces to precede the
                          22A0  ;               class title.
                          22A0  ;       R5 -    Used as a pointer into the class name strings.
                          22A0  ;       R6 -    Used to calculate the number of trailing spaces to follow the
                          22A0  ;               class title.
                          22A0  ;       R7 -    Points to descriptor containing the class name.
                          22A0  ;       R8 -    Used to calculate the number of spaces needed to center the
                          22A0  ;               class name.
                          22A0  ;       R9 -    Used to calculate the number of spaces needed to center the
                          22A0  ;               class name.
                          22A0  ;
                          22A0  ;       R11-    Used to index into the title length array.
                          22A0  ;
                          22A0  ;--
                          22A0
                          22A0  GET_CLASSES:
            0BFC 8F   BB  22A0          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R11>
                 5B   D4  22A4          CLRL    R11                     ; Set counter to 0
   53  0000159B'EF   DE  22A6          MOVAL   CLASS_BUF,R3            ; Get address of class buffer.
   56  00002222'EF   3C  22AD          MOVZWL  SCREEN_WIDTH,R6         ; Get screen width.
   57  000021D0'EF   DE  22B4          MOVAL   TITLE_LEN_ARRAY,R7      ; Get address of title length array.
   55  0000023B'EF   DE  22BB          MOVAL   CLASS_NAMES,R5          ; Get address of class descriptors.
           54   85   9A  22C2          MOVZBL  (R5)+,R4                ; Get number of bytes in first string.
                01   E0  22C5          BBS     #SCR$V_MORE_LFT,-       ; Branch if screen has been scrolled
     03 0000047A'EF       22C7                  SCROLL_FLAGS,2$         ;  left.
              014F   30  22CD          BSBW    DELM                    ; Add starting delimiter to line.
```

```
                          65  1F  22D0  6560 2$:     BLSSU   20$                       ; Branch if it won't fit on the line.
         52    00000001'EF  D0  22D2  6561           MOVL    CLASS_SELECT,R2           ; Get mask of classes to be displayed.
      00    52    01    5B  ED  22D9  6562 3$:      CMPZV   R11,#T,R2,#0              ; Check if class is enabled.
                          13  12  22DE  6563           BNEQ    5$                        ; Branch if so.
              54    FF   A5  9A  22E0  6564 4$:      MOVZBL  -1(R5),R4                 ; Get number of bytes
                    55   54  C0  22E4  6565           ADDL    R4,R5                     ; Point to next class string.
                    54   85  9A  22E7  6566           MOVZBL  (R5)+,R4                  ; Get number of bytes in string.
      FFE9 5B    01    08  F1  22EA  6567           ACBL    #CLS$V_ERR,#1,R11,3$      ; Branch if there's more classes to
                        0044  31  22F0  6568           BRW     20$                       ;  check.
                58      674B  D0  22F3  6569 5$:      MOVL    (R7)[R11],R8              ; Get length of the titles for this
                                  22F7  6570                                             ;  class.
                          58  D7  22F7  6571           DECL    R8                        ; Subtract one for the delimeter.
                          E5  19  22F9  6572           BLSS    4$                        ; Branch if no titles being displayed.
              59    58    54  C3  22FB  6573           SUBL3   R4,R8,R9                  ; Calculate the number of spaces
                                  22FF  6574                                             ;  to appear before and after the
                                  22FF  6575                                             ;  class name.
                          05  14  22FF  6576           BGTR    6$                        ; Branch if more characters in the
                                  2301  6577                                             ;  titles than in the class name.
                          59  D4  2301  6578           CLRL    R9                        ; Otherwise, no leading or trailing
                                  2303  6579                                             ;  spaces around the class name.
                    54    58  D0  2303  6580           MOVL    R8,R4                     ; Make length of class same as titles.
         59    59    FF  8F  9C  2306  6581 6$:      ROTL    #-1,R9,R9                 ; Calculate the number of leading
                                  230B  6582                                             ;  spaces to appear before the class
                                  230B  6583                                             ;  type in order to center it.
                        013B  30  230B  6584           BSBW    ADD_SPACES                ; Add leading spaces.
                          27  1F  230E  6585           BLSSU   20$                       ; Branch if the spaces won't fit on the
                                  2310  6586                                             ;  screen.
              05 59    1F  E5  2310  6587           BBCC    #31,R9,10$                ; Branch if even number of spaces.
                        012C  30  2314  6588           BSBW    ADD_ONE_SPACE             ; Add one more space.
                          1E  1F  2317  6589           BLSSU   20$                       ; Branch if it won't fit.
                          34  BB  2319  6590 10$:     PUSHR   #^M<R2,R4,R5>
           63    65    54  28  231B  6591           MOVC3   R4,(R5),(R3)              ; Add class type.
                          34  BA  231F  6592           POPR    #^M<R2,R4,R5>
                        0125  30  2321  6593           BSBW    ADD_SPACES                ; Add trailing spaces.
                          11  1F  2324  6594           BLSSU   20$                       ; Branch if the spaces won't fit on the
                                  2326  6595                                             ;  screen.
                        00F6  30  2326  6596           BSBW    DELM                      ; Add ending delimiter.
                          0C  1F  2329  6597           BLSSU   20$                       ; Branch if it won't fit.
          00002222'EF    56  B1  232B  6598 17$:     CMPW    R6,SCREEN_WIDTH           ; Will classes fit on the screen?
                                  2332  6599           JLEQU   4$                        ; Branch if they will.
00001593'EF    53    0000159B'8F  C3  2337  6600 20$:     SUBL3   #CLASS_BUF,R3,CLASS_DSCR; Save length of buffer.
                     0BFC 8F  BA  2343  6601           POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R11>
                          05  2347  6602           RSB
                              2348  6603
                              2348  6604
```

```
                              2348          .SBTTL  REFRESH routine.
                              2348  ;++
                              2348  ;
                              2348  ; FUNCTIONAL DESCRIPTION:
                              2348  ;
                              2348  ;       This routine causes the display header, which is not normally redraw on
                              2348  ;       each update, to be redraw.
                              2348  ;
                              2348  ; CALLING SEQUENCE:
                              2348  ;
                              2348  ;       CALLS   #0,REFRESH
                              2348  ;
                              2348  ; CALLED BY:
                              2348  ;
                              2348  ;       READ_LOOP
                              2348  ;       READ_WO_PROMPT
                              2348  ;       SHOW_CLUSTER
                              2348  ;       SHOW_LOCAL
                              2348  ;       ADD_FIELD
                              2348  ;       REM_FIELD
                              2348  ;       INCLUDE_CLASS
                              2348  ;       EXCLUDE_CLASS
                              2348  ;       SET_WIDTH
                              2348  ;       SET_SCREEN
                              2348  ;
                              2348  ; OUTPUTS:
                              2348  ;
                              2348  ;       RIP (Refresh in Progress) flag set at MISC_FLAGS.
                              2348  ;
                              2348  ;--
                              2348
                        0004  2348  .ENTRY  REFRESH,^M<R2>
                              234A          JBS     #MF$V_IN_OPEN,-         ; Skip refresh if executing from a
                              234A                  MISC_FLAGS,99$          ;  command procedure.
          000021AB'EF     94  2355          CLRB    SWITCH                  ; Initialize switch count.
00000476'EF    08         C8  235B          BISL    #MF$M_RIP,MISC_FLAGS    ; Indicate refresh in progress.
                          04  2362  99$:    RET
                              2363
                              2363
```

```
2363            .SBTTL  FIND_ENTRY routine.
2363    ;++
2363    ;
2363    ; FUNCTIONAL DESCRIPTION:
2363    ;
2363    ;       This routine is used to scan one of the class bitmaps for a set bit,
2363    ;       indicating that a field is to be displayed.  It returns the address
2363    ;       of the field descriptor to the calling routine.
2363    ;
2363    ;       This routine also performs the horizontal scrolling by ignoring the
2363    ;       first entries found as determined by the value stored in H_SCROLL_LINE.
2363    ;
2363    ; CALLING SEQUENCE:
2363    ;
2363    ;       JSB     FIND_ENTRY
2363    ;
2363    ; CALLED BY:
2363    ;
2363    ;       GET_TITLES
2363    ;       GET_DATA
2363    ;       ADD_FIELD
2363    ;
2363    ; IMPLICIT INPUTS:
2363    ;
2363    ;       ADRS_TABLE -    Contains the address of the table of entry addresses.
2363    ;       BITMAP_ADRS -   Contains the address of bitmap to be searched.
2363    ;       MAP_LEN -       Contains the length of the bitmap.
2363    ;       START_BIT -     Contains the number of the bit in the bitmap to start
2363    ;                       the search.
2363    ;                        (Should be zero on the first call to this routine.)
2363    ;       H_SCROLL_LINE   Set to the number of entries to skip before success is
2363    ;                       returned.
2363    ;
2363    ; OUTPUTS:
2363    ;
2363    ;       R10 =   Address of field descriptor.
2363    ;
2363    ; IMPLICIT OUTPUTS:
2363    ;
2363    ;       START_BIT updated to point to next bit to be checked.
2363    ;       MAP_LEN   updated to indicate the remaining length of the bitmap.
2363    ;
2363    ; RETURN STATUS:
2363    ;
2363    ;       R0 =    SS$_Normal if an entry is found.
2363    ;
2363    ; REGISTER USAGE:
2363    ;
2363    ;       R6  -   Pointer into the address table for the report.
2363    ;       R7  -   Pointer into the bitmap for the report.
2363    ;       R9  -   Used to determine the starting bit in the longword to scan.
2363    ;       R10 -   Used to determine the number of bits left to scan in the
2363    ;               longword.
2363    ;           -   Receives the address of the field descriptor once a entry has
2363    ;               been found.
2363    ;--
2363
```

```
                                     2363  FIND_ENTRY:
                 03C0 8F     BB      2363          PUSHR   #^M<R6,R7,R8,R9>
                       50    D4      2367          CLRL    R0                      ; Assume failure.
       56    00002264'EF     D0      2369          MOVL    ADRS_TABLE,R6           ; Get address of address table.
       57    00002268'EF     D0      2370          MOVL    BITMAP_ADRS,R7          ; Get address of bitmap.
       59    00002270'EF     D0      2377  5$:     MOVL    START_BIT,R9            ; Get the starting bit position.
                       5A    D4      237E          CLRL    R10                     ; Dividend must be a quadword.
   5A     51      59   20    7B      2380          EDIV    #32,R9,R1,R10           ; Calculate the number of bits left to
          5A      20   5A    C3      2385          SUBL3   R10,#32,R10             ;  scan in the longword.
   59     67      5A   59    EA      2389          FFS     R9,R10,(R7),R9          ; Scan for a set bit.
                       12    12      238E          BNEQ    20$                     ; Branch if found in this longword.
         0000226C'EF   51    D1      2390          CMPL    R1,MAP_LEN              ; Any more map left to scan?
                       32    13      2397          BEQL    99$                     ; Branch if not.
         00002270'EF   59    D0      2399          MOVL    R9,START_BIT            ; Save starting bit for next longword.
                       D5    11      23A0          BRB     5$                      ; Scan next longword.
              5A     6649    D0      23A2  20$:    MOVL    (R6)[R9],R10            ; Get the address from the table.
   00002270'EF     59  01    C1      23A6          ADDL3   #1,R9,START_BIT         ; Save starting bit for next
                                     23AE                                          ;  call to FIND_ENTRY.
                    0C AA    D3      23AE          BITL    FD$L_CLASS(R10),-       ; Is this entry in a selected class?
             00000001'EF             23B1                  CLASS_SELECT            ;
                       BF    13      23B6          BEQLU   5$                      ; Branch if not.
             0000046C'EF     D5      23B8          TSTL    H_SCROLL_LINE           ; Is this field scrolled off the left of
                                     23BE                                          ;  the screen?
                       08    19      23BE          BLSS    30$                     ; Branch if not.
             0000046C'EF     D7      23C0          DECL    H_SCROLL_LINE           ; Count down the number of fields to be
                                     23C6                                          ;    ignored.
                       AF    18      23C6          BGEQ    5$                      ; Ignore this field.
                 50    01    D0      23C8  30$:    MOVL    #SS$_NORMAL,R0          ; Indicate success.
                 03C0 8F     BA      23CB  99$:    POPR    #^M<R6,R7,R8,R9>        ;
                             05      23CF          RSB                             ;
                                     23D0
                                     23D0
```

```
                                 23D0          .SBTTL  Miscellaneous routines.
                                 23D0      ;++
                                 23D0      ;
                                 23D0      ; FUNCTIONAL DESCRIPTION:
                                 23D0      ;
                                 23D0      ;       Routine to clear the terminal's screen.
                                 23D0      ;
                                 23D0      ; CALLING SEQUENCE:
                                 23D0      ;
                                 23D0      ;       JSB     CLEAR_SCREEN
                                 23D0      ;
                                 23D0      ; CALLED BY:
                                 23D0      ;
                                 23D0      ;       MAIN_ERR
                                 23D0      ;       SHOW_CLUSTER
                                 23D0      ;       SHOW_LOCAL
                                 23D0      ;
                                 23D0      ; IMPLICIT OUTPUTS:
                                 23D0      ;
                                 23D0      ;       COL_NUM (column number) and LINE_NUM (line number) are set to 1
                                 23D0      ;
                                 23D0      ; RETURN STATUS:
                                 23D0      ;
                                 23D0      ;       R0 = SS$_NORMAL or failing status from SRCPKG routine.
                                 23D0      ;
                                 23D0      ;
                                 23D0      ;--
                                 23D0
                                 23D0      CLEAR_SCREEN:
                 50    01    D0  23D0              MOVL    #SS$_NORMAL,R0          ; Assume success.
       0000224C'EF    01    D0  23D3              MOVL    #1,COL_NUM              ; Start at column 1
       00002248'EF    01    D0  23DA              MOVL    #1,LINE_NUM             ;  and line 1.
    13 00002247'EF    02    E0  23E1              BBS     #DVT$V_HARD,OUT_DEV,99$ ; Skip if output is hardcopy device.
          0000224C'EF    3F  23E9              PUSHAW  COL_NUM                 ; Clear the screen.
          00002248'EF    3F  23EF              PUSHAW  LINE_NUM                ;
       00000000'GF    02    FB  23F5              CALLS   #2,G^LIB$ERASE_PAGE     ;
                       05  23FC      99$:    RSB                             ; Return with status.
                                 23FD
                                 23FD
```

```
                                23FD  ;++
                                23FD  ;
                                23FD  ; FUNCTIONAL DESCRIPTION:
                                23FD  ;
                                23FD  ;       Routines to erase one or two lines on the terminal.
                                23FD  ;
                                23FD  ; CALLING SEQUENCE:
                                23FD  ;
                                23FD  ;       JSB     ERASE_LINE
                                23FD  ;       JSB     ERASE_TWO_LINES
                                23FD  ;
                                23FD  ; CALLED BY:
                                23FD  ;
                                23FD  ;       ERASE_PAGE
                                23FD  ;       DRAW_LINE
                                23FD  ;
                                23FD  ;       READ_WO_PROMPT
                                23FD  ;       DISPLAY_PTP
                                23FD  ;
                                23FD  ; IMPLICIT OUTPUTS:
                                23FD  ;
                                23FD  ;       LINE_NUM (line number) is incremented.
                                23FD  ;
                                23FD  ; RETURN STATUS:
                                23FD  ;
                                23FD  ;       R0 = SS$_NORMAL or failing status from SRCPKG routine.
                                23FD  ;
                                23FD  ;
                                23FD  ;--
                                23FD          .ENABL  LSB
                                23FD
                                23FD  ERASE_TWO_LINES:
                  03    10      23FD          BSB     ERASE_LINE              ; Erase a line.
               1C 50    E9      23FF          BLBC    R0,99$                  ; Branch on error.
                                2402
                                2402  ERASE_LINE:
               50 01    D0      2402          MOVL    #SS$_NORMAL,R0          ; Assume success.
         0000224C'EF    3F      2405          PUSHAW  COL_NUM                 ; Erase the line
         00002248'EF    3F      240B          PUSHAW  LINE_NUM                ;
      00000000'GF 02    FB      2411          CALLS   #2,G^LIB$ERASE_LINE     ;
         00002248'EF    D6      2418          INCL    LINE_NUM                ; Bump line number.
                  05            241E  99$:    RSB                             ; Return with status.
                                241F
                                241F          .DSABL  LSB
                                241F
```

```
                                    241F  ;++
                                    241F  ;
                                    241F  ; FUNCTIONAL DESCRIPTION:
                                    241F  ;
                                    241F  ;       The following routine adds a control string to a line of data to
                                    241F  ;        indicate the start, and end of fields in the data line.
                                    241F  ;
                                    241F  ; CALLING SEQUENCE:
                                    241F  ;
                                    241F  ;       JSB     DELM
                                    241F  ;
                                    241F  ; CALLED BY:
                                    241F  ;
                                    241F  ;       GET_TITLES
                                    241F  ;       GET_CLASSES
                                    241F  ;
                                    241F  ; INPUTS:
                                    241F  ;
                                    241F  ;       R3 -    Address of the next free character position in the buffer
                                    241F  ;       R6 -    Number of bytes that can be added to the buffer before its
                                    241F  ;               length exceeds that of the screen line.
                                    241F  ;
                                    241F  ; OUTPUTS:
                                    241F  ;
                                    241F  ;       R3 -    Next free position in the buffer.
                                    241F  ;       R6 -    Updated number of bytes that can be added to the buffer before
                                    241F  ;               its length exceeds that of the screen line.
                                    241F  ;
                                    241F  ;
                                    241F  ; RETURN STATUS:
                                    241F  ;
                                    241F  ;       R0 =    SS$_NORMAL.
                                    241F  ;
                                    241F  ;--
                                    241F
                                    241F  DELM:
                          34   BB   241F          PUSHR   #^M<R2,R4,R5>
        06 00002247'EF    01   E0   2421          BBS     #DVT$V_DCRT,OUT_DEV,10$ ; Branch if terminal is a DECcrt.
              83    7C 8F  90   2429          MOVB    #^A/|/,(R3)+            ; Store a VT52 vertical bar "|".
                          0C   11   242D          BRB     20$                     ; Jump to common code.
   0000017B'EF   00000173'EF   28   242F  10$:    MOVC3   DELM_DSCR,DELM_DSCR+8,- ; Store a DECcrt vertical bar.
                          63        243A                  (R3)                    ;
                          56   D7   243B  20$:    DECL    R6                      ; Subtract the length of the string
                                    243D                                          ;  in characters as it will appear on
                                    243D                                          ;  the screen.
                          34   BA   243D          POPR    #^M<R2,R4,R5>
                     50   01   D0   243F          MOVL    #SS$_NORMAL,R0
                               05   2442          RSB
                                    2443
                                    2443
```

```
                    2443  ;++
                    2443  ;
                    2443  ; FUNCTIONAL DESCRIPTION:
                    2443  ;
                    2443  ;       This routine adds a single space to a buffer at the location pointed to
                    2443  ;       by R3.
                    2443  ;
                    2443  ; CALLING SEQUENCE:
                    2443  ;
                    2443  ;       JSB     ADD_ONE_SPACE
                    2443  ;
                    2443  ; CALLED BY:
                    2443  ;
                    2443  ;       GET_TITLES
                    2443  ;       GET_CLASSES
                    2443  ;
                    2443  ; INPUTS:
                    2443  ;
                    2443  ;       R3 -    Points to the next free position in the buffer.
                    2443  ;       R6 -    Contains the number of characters that can be output to the
                    2443  ;               buffer before the number of characters in the buffer exceeds
                    2443  ;               the width of the screen.
                    2443  ;
                    2443  ; OUTPUTS:
                    2443  ;
                    2443  ;       Single space stored in buffer pointed to by R3.
                    2443  ;
                    2443  ; IMPLICIT OUTPUTS:
                    2443  ;
                    2443  ;       R3 and R6 updated
                    2443  ;
                    2443  ; RETURN STATUS:
                    2443  ;
                    2443  ;       None.
                    2443  ;
                    2443  ;--
                    2443
                    2443  ADD_ONE_SPACE:
  83    20    90    2443          MOVB    #^A/ /,(R3)+            ; Add one space to the buffer.
        56    D7    2446          DECL    R6                      ; Decrease room left on display line.
              05    2448          RSB
                    2449
```

```
                     2449  ;++
                     2449  ;
                     2449  ; FUNCTIONAL DESCRIPTION:
                     2449  ;
                     2449  ;         This routine adds a number of spaces to a buffer by recusively calling
                     2449  ;         ADD_ONE_SPACE.
                     2449  ;
                     2449  ; CALLING SEQUENCE:
                     2449  ;
                     2449  ;         JSB     ADD_SPACES
                     2449  ;
                     2449  ; CALLED BY:
                     2449  ;
                     2449  ;         GET_TITLES
                     2449  ;         GET_CLASSES
                     2449  ;
                     2449  ; INPUTS:
                     2449  ;
                     2449  ;         R9 -    Number of spaces to output.
                     2449  ;         R3 -    Points to the next free position in the buffer.
                     2449  ;         R6 -    Contains the number of characters that can be output to the
                     2449  ;                 buffer before the number of characters in the buffer exceeds
                     2449  ;                 the width of the screen.
                     2449  ;
                     2449  ; OUTPUTS:
                     2449  ;
                     2449  ;         Spaces stored in buffer pointed to by R3.
                     2449  ;
                     2449  ; IMPLICIT OUTPUTS:
                     2449  ;
                     2449  ;         R3 and R6 updated
                     2449  ;
                     2449  ; RETURN STATUS:
                     2449  ;
                     2449  ;         None.
                     2449  ;
                     2449  ;--
                     2449
                     2449  ADD_SPACES:
0200 8F          BB  2449          PUSHR   #^M<R9>
     59          95  244D          TSTB    R9                      ; Any spaces to be output?
     05          12  244F          BNEQ    10$                     ; Branch if so.
59   01          D0  2451          MOVL    #1,R9                   ; Otherwise clear zero flag.
     0B          11  2454          BRB     99$                     ; And return.
     EB          10  2456  10$:    BSB     ADD_ONE_SPACE           ; Output one space.
     07          1F  2458          BLSSU   99$                     ; Return if there's no more room on the
                     245A                                          ;  line.
     59          97  245A          DECB    R9                      ; Count down the number of spaces to
                     245C                                          ;  output.
     F8          12  245C          BNEQ    10$                     ; Loop till all spaces are output.
59   01          D0  245E          MOVL    #1,R9                   ; Clear zero flag.
0200 8F          BA  2461  99$:    POPR    #^M<R9>
                 05  2465          RSB
                     2466
                     2466
```

```
                              2466  ;++
                              2466  ;
                              2466  ; FUNCTIONAL DESCRIPTION:
                              2466  ;
                              2466  ;       This routine outputs a line to the screen buffer.  If the line causes
                              2466  ;       the screen buffer to overflow then the screen buffer is output to the
                              2466  ;       screen and the line output retried.
                              2466  ;
                              2466  ;       This routine also handles the case where the output line is longer than
                              2466  ;       the width of the screen by outputing a trunction character at the end
                              2466  ;       of the line.  The truncation character is either a block or a diamond
                              2466  ;       shape depending on terminal type.
                              2466  ;
                              2466  ; CALLING SEQUENCE:
                              2466  ;
                              2466  ;       JSB     DRAW_LINE
                              2466  ;
                              2466  ; CALLED BY:
                              2466  ;
                              2466  ;       DRAW_LAST_LINE
                              2466  ;       GET_HEADER
                              2466  ;       GET_DATA
                              2466  ;
                              2466  ; INPUTS:
                              2466  ;
                              2466  ;       Address of output line descriptor in R11.
                              2466  ;
                              2466  ; IMPLICIT OUTPUTS:
                              2466  ;
                              2466  ;       LINE_NUM = LINE_NUM + 1
                              2466  ;
                              2466  ; RETURN STATUS:
                              2466  ;
                              2466  ;       R0 =    SS$_NORMAL.
                              2466  ;
                              2466  ;--
                              2466
                              2466  DRAW_LINE:
              0000224C'EF  3F 2466          PUSHAW  COL_NUM                 ; Output the line to the screen buffer.
              00002248'EF  3F 246C          PUSHAW  LINE_NUM                ;
                       5B  DD 2472          PUSHL   R11                     ;
          00000000'GF  03  FB 2474          CALLS   #3,G^LIB$PUT_SCREEN     ;
                  0F   50  E8 247B          BLBS    R0,10$                  ; Branch if no error.
          00000000'GF  00  FB 247E          CALLS   #0,G^LIB$PUT_BUFFER     ; Output the screen buffer to the
                              2485                                          ;  screen.
                              2485          JLBC    R0,99$                  ; Branch on error.
                       D9  11 248B          BRB     DRAW_LINE               ; Try outputing the line again.
0000224C'EF   01  000021C4'EF  A1 248D  10$:  ADDW3   HEADER_SIZE,#1,COL_NUM  ; Erase the remainer of the line.
                     FF66  30 2499          BSBW    ERASE_LINE              ;
                              249C          JLBC    R0,99$                  ; Branch on error.
    00002222'EF   0000224C'EF  B1 24A2          CMPW    COL_NUM,SCREEN_WIDTH    ; Does the line fit on the screen?
                       19  14 24AD          BGTR    20$                     ; Branch if not.
                       01  8A 24AF          BICB    #SCR$M_MORE_RT,-        ; Clear the flag that indicates that
              0000047A'EF     24B1                  SCROLL_FLAGS            ;  there is more data off the right of
                              24B6                                          ;  the screen.
                       93     24B6          BITB    #<SCR$M_MORE_LFT!-      ; Are any of the other scroll flags
                              24B7                  SCR$M_MORE_UP!-         ;  set?
```

```
                                   24B7               SCR$M_MORE_DWN>,-              ;
          0000047A'EF   0E         24B7               SCROLL_FLAGS                   ;
                        10   12    24BD       BNEQ    40$                            ; Branch if not.
          0000047A'EF   10   8A    24BF       BICB    #SCR$M_MORE,SCROLL_FLAGS;      ; Otherwise, be sure to clear the
                                   24C6                                              ;  scroll summary flag.
                        07   11    24C6       BRB     40$                            ; Jump to common.
                        11   88    24C8 20$:  BISB    #SCR$M_MORE_RT!SCR$M_MORE,-    ; Set the scroll flag to indicate
               0000047A'EF         24CA               SCROLL_FLAGS                   ;  that there is more data off the
                                   24CF                                              ;  right of the screen.
          00000476'EF   04   E0    24CF 40$:  BBS     #MF$V_CONT,MISC_FLAGS,-        ; Branch if continuous display.
                        41         24D6               99$
       39 00002247'EF   01   E1    24D7       BBC     #DVT$V_DCRT,OUT_DEV,99$        ; Branch if terminal isn't a DECcrt.
00002258'EF    00002248'EF   D1    24DF       CMPL    LINE_NUM,LAST_DATA_LINE        ; Have we hit the bottom?
                        2C   12    24EA       BNEQ    99$                            ; Branch if not.
                      F4ED   30    24EC       BSBW    SET_REGION_DATA                ; Set scrolling region to include only
                                   24EF                                              ;  the data area of the screen.
               0000224C'EF   3F    24EF       PUSHAW  COL_NUM                        ; Set cursor back to last line.
               00002248'EF   3F    24F5       PUSHAW  LINE_NUM                       ;
          00000000'GF   02   FB    24FB       CALLS   #2,G*LIB$SET_CURSOR
                     13 50   E9    2502       BLBC    R0,99$                         ; Branch on error.
          00000000'GF   00   FB    2505       CALLS   #0,G^LIB$UP_SCROLL             ; Scroll up one line.
                     09 50   E9    250C       BLBC    R0,99$                         ; Branch on error.
                      F4BC   30    250F       BSBW    SET_REGION_FULL                ; Reset the scrolling region to use the
                                   2512                                              ;  full screen.
               00002248'EF   D7    2512       DECL    LINE_NUM                       ; Adjust line number.
          0000224C'EF   01   D0    2518 99$:  MOVL    #1,COL_NUM                     ; Reset column counter to first column.
                             05    251F       RSB
                                   2520
                                   2520
```

```
                                        2520  ;++
                                        2520  ;
                                        2520  ; FUNCTIONAL DESCRIPTION:
                                        2520  ;
                                        2520  ;       This routine is used to get a line of data with all the fields filled
                                        2520  ;       with spaces.  This is done by disabling the display of all classes and
                                        2520  ;       calling the GET_DATA routine.
                                        2520  ;
                                        2520  ; CALLING SEQUENCE:
                                        2520  ;
                                        2520  ;       JSB     GET_BLANK
                                        2520  ;
                                        2520  ; CALLED BY:
                                        2520  ;
                                        2520  ;       CLUSTER_REPORT
                                        2520  ;
                                        2520  ; OUTPUTS:
                                        2520  ;
                                        2520  ;       One line of data output to the screen buffer.
                                        2520  ;
                                        2520  ; RETURN STATUS:
                                        2520  ;
                                        2520  ;       None.
                                        2520  ;
                                        2520  ; REGISTER USAGE:
                                        2520  ;
                                        2520  ;       R4 -    Used to save a mask of currently enabled classes.
                                        2520  ;
                                        2520  ;--
                                        2520
                                        2520  GET_BLANK:
08 0000043C'EF    00002248'EF    E2     2520          BBSS    LINE_NUM,BLANK_LINES,10$; Indicate that this is a blank line.
       14 00000476'EF    03      E0     252C          BBS     #MF$V_RIP,MISC_FLAGS,20$; If refresh in progress, make sure
                                        2534                                          ;  we draw all the lines.
            00002248'EF          E1     2534  10$:    BBC     LINE_NUM,-              ; Branch this line wasn't blank
         08 00000440'EF                 253A          PRIOR_BLANK_LINES,20$           ;  during the last pass.
            00002248'EF          D6     2540          INCL    LINE_NUM                ; Otherwise just bump the line number.
                         1D      11     2546          BRB     99$                     ; And finish up.
            000021F8'EF          DD     2548  20$:    PUSHL   DATA_CNTRL              ; Save a copy of the current classes.
            000021F8'EF          D4     254E          CLRL    DATA_CNTRL              ; Disable all classes.
         00002729'EF     00      FB     2554          CALLS   #0,GET_DATA             ; Get blank line of data.
                      07 50      E9     255B          BLBC    R0,99$                  ; Branch on error.
            000021F8'EF        8ED0     255E          POPL    DATA_CNTRL              ; Reenable classes.
                                 05     2565  99$:    RSB
                                        2566
                                        2566
```

```
                               2566  ;++
                               2566  ;
                               2566  ; FUNCTIONAL DESCRIPTION:
                               2566  ;
                               2566  ;         This routine simply calls the cluster display set up routine if
                               2566  ;         the local ports report is being displayed or it calls the local port
                               2566  ;         display set up routine if the cluster report is being displayed.
                               2566  ;
                               2566  ; CALLING SEQUENCE:
                               2566  ;
                               2566  ;         JSB     SWITCH_DISPLAY
                               2566  ;
                               2566  ; CALLED BY:
                               2566  ;
                               2566  ;
                               2566  ;--
                               2566
                               2566  SWITCH_DISPLAY:
                               2566
                  00     E0    2566          BBS     #RPT$V_CLSTR,-          ; Branch if we're displaying the
     09 00000000'EF            2568                  CURRENT_REPORT,10$      ;  cluster report.
 00000929'EF      00     FB    256E          CALLS   #0,SHOW_CLUSTER         ; Display the cluster report.
                  07     11    2575          BRB     99$                     ; Jump to common code.
 0000099F'EF      00     FB    2577  10$:    CALLS   #0,SHOW_LOCAL           ; Display the local report.
                         05    257E  99$:    RSB
                               257F
                               257F
```

```
257F            .SBTTL  Data formatting routines.
257F    ;++
257F    ;
257F    ; FUNCTIONAL DESCRIPTION:
257F    ;
257F    ;       This routine formats the display data for the GET_DATA routine.  It
257F    ;       returns the formatted data in FORMAT_BUF and the number of characters
257F    ;       worth of display data in FORMAT_BUF_LEN.
257F    ;
257F    ;       The field descriptor parameter FD$B_FORMAT determines if the routine
257F    ;       will format the output in ascii, hexadecimal, or decimal.  Up to a
257F    ;       quadword of data can be formatted in decimal.  Ascii and hexadecimal
257F    ;       formatting has no restrictions.
257F    ;
257F    ; CALLING SEQUENCE:
257F    ;
257F    ;       JSB     FORMAT_DATA
257F    ;
257F    ; CALLED BY:
257F    ;
257F    ;       DISPLAY_PTP
257F    ;       OUTPUT_DDM
257F    ;       GET_DATA
257F    ;       DISPLAY_FIELD
257F    ;       DISPLAY_STRING
257F    ;       DISPLAY_CIRCUITS
257F    ;       DISPLAY_INCN_TIME
257F    ;       DISPLAY_PROC_NAME
257F    ;       DISPLAY_PORT_STATUS
257F    ;       DISPLAY_PORT_TYPE
257F    ;
257F    ; INPUTS:
257F    ;
257F    ;       R10     = Address of field descriptor.
257F    ;
257F    ; OUTPUTS:
257F    ;
257F    ;       Formated data string returned in FORMAT_BUF.
257F    ;       Length of formatted data string returned in FORMAT_BUF_LEN.
257F    ;
257F    ;
257F    ; RETURN STATUS:
257F    ;
257F    ;       R0 =    SS$_NORMAL or failed status.
257F    ;
257F    ;
257F    ; REGISTER USAGE:
257F    ;
257F    ;       R3 -    Points to the next free position in FORMAT_BUF where the
257F    ;               the conversion is temporarily stored.
257F    ;       R5 -    Contains the byte returned from the CONVERT_DEC routine.
257F    ;       R6 -    Used to index into the data.
257F    ;       R7 -    Points to the address of the data.
257F    ;       R8 -    Used by HEX_DISPLAY to convert a byte to two hex digits.
257F    ;          -    Used by DEC_DISPLAY to hold value from conversion table.
257F    ;       R10 -   Points to the field descriptor.
257F    ;       R11 -   Used to calculate the length of the conversion.
```

```
                                257F  ;       R10,R11 Used by DEC_DISPLAY to hold the quadword of data to be
                                257F  ;               converted to decimal.
                                257F  ;
                                257F  ;--
                                257F
                                257F  FORMAT_DATA:
           0DE8 8F       BB     257F          PUSHR   #^M<R3,R5,R6,R7,R8,R10,R11>
                50       D4     2583          CLRL    R0                          ; Assume failure.
 53     0000215B'EF      DE     2585          MOVAL   FORMAT_BUF,R3               ; Get address of temporary buffer.
                                258C          CASE    FD$B_FORMAT(R10),-          ; Dispatch on format type.
                                258C                  <ASCII_DISPLAY,-
                                258C                  HEX_DISPLAY,-
                                258C                  DEC_DISPLAY,-
                                258C                  ASCIC_DISPLAY,-
                                258C                  TIME_DISPLAY>,TYPE=B,LIMIT=#1
                                259B
                                259B
                                259B  TIME_DISPLAY:
                                259B          $ASCTIM_S -                         ; Convert time to ascii.
                                259B                  TIMBUF=ASC_TIM_DSCR,-       ;
                                259B                  TIMADR=(R7)                 ;
                                25AE  ;
                                25AE  ; Shift the last tenth through eighteenth characters of the time string
                                25AE  ; left two places.  This is to condense the date by effectively removing
                                25AE  ; the first two digits of the year, i.e., dd-mmm-yyyy hh:mm:ss.cc becomes
                                25AE  ; dd-mmm-yy hh:mmmm:ss.cc.  The last eigth characters are simply not output,
                                25AE  ; leaving dd-mmm-yy hh:mm.
                                25AE  ;
                                25AE
 51     00001F9F'EF      DE     25AE          MOVAL   TIMBUF+9,R1                 ; Get address of "yyyy"
 57     00001F9D'EF      DE     25B5          MOVAL   TIMBUF+7,R7                 ; Get address of "yy h"
             87   81     7D     25BC          MOVQ    (R1)+,(R7)+                 ; Shift first four characters left 2.
             87   81     7D     25BF          MOVQ    (R1)+,(R7)+                 ; Shift next four characters left 2.
 57     00001F96'EF      DE     25C2          MOVAL   TIMBUF,R7                   ; Get address of time string.
                                25C9                                              ; Fall through to ASCII_DISPLAY.
                                25C9
                                25C9
                                25C9  ;
                                25C9  ; Format the data in ASCII.
                                25C9  ;
                                25C9
                                25C9  ASCII_DISPLAY:
                  56     D4     25C9          CLRL    R6                          ; Zero index.
               04 AA     9A     25CB          MOVZBL  FD$B_LENGTH(R10),-          ; Save length of string for return.
        00002153'EF             25CE                  FORMAT_BUF_LEN              ;
          83    6746     90     25D3  10$:    MOVB    (R7)[R6],(R3)+              ; Store a character.
                  56     96     25D7          INCB    R6                          ; Bump index.
          04 AA   56     91     25D9          CMPB    R6,FD$B_LENGTH(R10)         ; Are we done?
                  F4     12     25DD          BNEQU   10$                         ; Loop if not.
             20   73     91     25DF  20$:    CMPB    -(R3),#^A/ /                ; Any trailing spaces?
                  07     12     25E2          BNEQU   30$                         ; Branch if not.
     F4 00002153'EF      F5     25E4          SOBGTR  FORMAT_BUF_LEN,20$          ; Remove trailing space.
                  53     D6     25EB  30$:    INCL    R3                          ; Restore R3 to last character plus 1.
                0ODF     30     25ED          BSBW    CHECK_VALID                 ; Make sure the string contains valid
                                25F0                                              ;  ascii characters.
                                25F0          JLBC    R0,FERR                     ; Branch if string contains non-ascii.
                00CE     31     25F6          BRW     FDONE                       ; Done.
```

```
                             25F9
                             25F9  ;
                             25F9  ; Format the data in HEXADECIMAL
                             25F9  ;
                             25F9
                             25F9  HEX_DISPLAY:
        56   04 AA     9A    25F9          MOVZBL  FD$B_LENGTH(R10),R6     ; Get the number of bytes of data.
     5B   56   01      9C    25FD          ROTL    #1,R6,R11               ; Get the maximum length of the
                             2601                                          ;  converted string.
               56      D7    2601  10$:    DECL    R6                      ; Count down number of bytes.
               0B      19    2603          BLSS    15$                     ; Branch if data equals 0.
        58   6746      9A    2605          MOVZBL  (R7)[R6],R8             ; Get a byte of data.
               16      12    2609          BNEQ    20$                     ; Branch if not 2 leading zeros.
          5B   02      C2    260B          SUBL    #2,R11                  ; Subtract 2 leading zeros from the
                             260E                                          ;  length of the converted string.
               F1      11    260E          BRB     10$                     ; Loop for all leading zeros.
 00002153'EF   01      D0    2610  15$:    MOVL    #1,FORMAT_BUF_LEN       ; Length equals 1.
 0000215B'EF   30      90    2617          MOVB    #^A/0/,FORMAT_BUF       ; String equals "0"
             00A6      31    261E          BRW     FDONE                   ; Done.
   58   58   FC 8F     78    2621  20$:    ASHL    #-4,R8,R8               ; Scratch lower nyble.
               04      12    2626          BNEQ    HIGH_NYBLE              ; Branch if not single leading zero.
               5B      D7    2628          DECL    R11                     ; Subtract one leading zero from the
                             262A                                          ;  length of the converted string.
               0C      11    262A          BRB     LOW_NYBLE               ; Start format with low nyble.
                             262C  HIGH_NYBLE:
        58   6746      9A    262C          MOVZBL  (R7)[R6],R8             ; Get a byte of data.
   58   58   FC 8F     9C    2630          ROTL    #-4,R8,R8               ; Scratch the low nyble.
             00C4      30    2635          BSBW    CONVERT_HEX             ; Convert the nyble to ascii.
                             2638  LOW_NYBLE:
        58   6746      9A    2638          MOVZBL  (R7)[R6],R8             ; Get full byte back.
 58   000000F0 8F      CA    263C          BICL    #^X000000F0,R8          ; Mask the low nyble.
             00B6      30    2643          BSBW    CONVERT_HEX             ; Convert the nyble to ascii.
            E3 56      F4    2646          SOBGEQ  R6,HIGH_NYBLE           ; Loop for entire value.
 00002153'EF   5B      D0    2649          MOVL    R11,FORMAT_BUF_LEN      ; Save length of conversion.
             0074      31    2650          BRW     FDONE                   ; Done.
                             2653
                             2653
                             2653  ;
                             2653  ; Format the data in DECIMAL.
                             2653  ;
                             2653
                             2653  DEC_DISPLAY:
        56   04 AA     9A    2653          MOVZBL  FD$B_LENGTH(R10),R6     ; Get the number of bytes of data.
          08   56      91    2657          CMPB    R6,#8                   ; We can't handle more than 8 bytes.
               6E      14    265A          BGTR    FERR                    ; Error if more than 8 bytes.
               5A      7C    265C          CLRQ    R10                     ; Start with quadword of zero.
               56      D7    265E  10$:    DECL    R6                      ; Count down the bytes.
               0A      19    2660          BLSS    15$                     ; Branch if we've got all the bytes.
     5A   5A   08      79    2662          ASHQ    #8,R10,R10              ; Shift the byte into quadword.
        5A   6746      90    2666          MOVB    (R7)[R6],R10            ; Get a byte of data.
               F2      11    266A          BRB     10$                     ; Loop for entire value.
 57   000004CA'EF      DE    266C  15$:    MOVAL   DEC_TAB,R7              ; Point to conversion table.
          56   14      D0    2673          MOVL    #DEC_TAB_LEN,R6         ; 20 conversions steps in all.
          58   87      7D    2676  20$:    MOVQ    (R7)+,R8                ; Get a value from the conversion
                             2679                                          ;  table.
             0098      30    2679          BSBW    CONVERT_DEC             ; Convert to one decimal digit.
 0000215B'8F   53      D1    267C          CMPL    R3,#FORMAT_BUF          ; Have any characters been output yet?
```

```
                       04   12 2683  7305          BNEQU   30$                          ; Branch if so.
                       55   95 2685  7306          TSTB    R5                           ; Was the conversion result = 0.
                       04   13 2687  7307          BEQLU   40$                          ; If so, don't save the character.
             83   55   30   81 2689  7308 30$:     ADDB3   #^X30,R5,(R3)+               ; Convert to ascii and save character.
                    E6 56   F5 268D  7309 40$:     SOBGTR  R6,20$                       ; Loop for all 20 steps.
00002153'EF   53  0000215B'8F C3 2690  7310          SUBL3   #FORMAT_BUF,R3,-             ; Save the length of the conversion.
                               269C  7311                  FORMAT_BUF_LEN               ;
                       0E   12 269C  7312          BNEQ    50$                          ; Branch if result wasn't zero.
          0000215B'EF  30   90 269E  7313          MOVB    #^A/0/,FORMAT_BUF            ; Store a "0"
          00002153'EF  01   D0 26A5  7314          MOVL    #1,FORMAT_BUF_LEN            ; Length equals one byte.
                     0018   31 26AC  7315 50$:     BRW     FDONE                        ; Done.
                               26AF  7316
                               26AF  7317 ASCIC_DISPLAY:
          00002153'EF  87   9A 26AF  7318          MOVZBL  (R7)+,FORMAT_BUF_LEN         ; Save length of string for return.
                       34   BB 26B6  7319          PUSHR   #^M<R2,R4,R5>
     63   67  00002153'EF   28 26B8  7320          MOVC3   FORMAT_BUF_LEN,(R7),(R3); Save the string.
                       34   BA 26C0  7321          POPR    #^M<R2,R4,R5>
                       0B   10 26C2  7322          BSB     CHECK_VALID                  ; Make sure string contains valid ascii
                               26C4  7323                                               ;  characters.
                    03 50   E9 26C4  7324          BLBC    R0,FERR                      ; Branch if not.
                               26C7  7325
                  50   01   D0 26C7  7326 FDONE:   MOVL    #SS$_NORMAL,R0               ; Indicate success.
                  0DE8 8F   BA 26CA  7327 FERR:    POPR    #^M<R3,R5,R6,R7,R8,R10,R11>
                       05      26CE  7328          RSB
                               26CF  7329
```

```
                              26CF  ;++
                              26CF  ;
                              26CF  ; FUNCTIONAL DESCRIPTION:
                              26CF  ;
                              26CF  ;       This routine checks the string in the format buffer for valid ascii
                              26CF  ;       characters.
                              26CF  ;
                              26CF  ; CALLING SEQUENCE:
                              26CF  ;
                              26CF  ;       JSB     CHECK_VALID
                              26CF  ;
                              26CF  ; CALLED BY:
                              26CF  ;
                              26CF  ;       ASCII_DISPLAY
                              26CF  ;       ASCIC_DISPLAY
                              26CF  ;
                              26CF  ; INPUTS:
                              26CF  ;
                              26CF  ;       FORMAT_BUF contains the string.
                              26CF  ;       FORMAT_BUF_LEN contains the length of the string.
                              26CF  ;
                              26CF  ; RETURN STATUS:
                              26CF  ;
                              26CF  ;       R0 = SS$_NORMAL if string contains valid ascii characters.
                              26CF  ;
                              26CF  ; REGISTER USAGE:
                              26CF  ;
                              26CF  ;       R2 -    Counts down the characters in the string.
                              26CF  ;       R3 -    Points into the buffer to the character to check.
                              26CF  ;
                              26CF  ;--
                              26CF
                              26CF  CHECK_VALID:
                   0C    BB   26CF          PUSHR   #^M<R2,R3>
                   50    D4   26D1          CLRL    R0                      ; Assume failure.
     52  00002153'EF     D0   26D3          MOVL    FORMAT_BUF_LEN,R2       ; Get length of string.
     53  0000215B'EF     DE   26DA          MOVAL   FORMAT_BUF,R3           ; Get address of format buffer.
                   03    10   26E1          BSB     CHECK_ASCII             ; Check for none ascii characters.
                   0C    BA   26E3          POPR    #^M<R2,R3>
                         05   26E5          RSB
                              26E6
                              26E6
```

```
                     26E6  ;++
                     26E6  ;
                     26E6  ; FUNCTIONAL DESCRIPTION:
                     26E6  ;
                     26E6  ;       Check a string for valid ascii characters.  Return immediately if an
                     26E6  ;       invalid ascii character is found with R3 pointing to the character and
                     26E6  ;       failed status in R0.
                     26E6  ;
                     26E6  ; CALLING SEQUENCE:
                     26E6  ;
                     26E6  ;       JSB     CHECK_ASCII
                     26E6  ;
                     26E6  ; CALLED BY:
                     26E6  ;
                     26E6  ;       CHECK_VALID
                     26E6  ;       BRDCST
                     26E6  ;
                     26E6  ; INPUTS:
                     26E6  ;
                     26E6  ;       R2 = Length of string
                     26E6  ;       R3 = Address of string
                     26E6  ;
                     26E6  ; RETURN STATUS:
                     26E6  ;
                     26E6  ;       If all charcters in the string are valid ascii:
                     26E6  ;
                     26E6  ;               R0 = SS$_NORMAL
                     26E6  ;               R2 = 0
                     26E6  ;               R3 = pointer to character following the string.
                     26E6  ;
                     26E6  ;       If invalid character found:
                     26E6  ;
                     26E6  ;               R0 = failed status
                     26E6  ;               R2 = number of characters remaining in string.
                     26E6  ;               R3 = points to failed character.
                     26E6  ;
                     26E6  ;--
                     26E6
                     26E6  CHECK_ASCII:
            50   D4  26E6          CLRL    R0                      ; Assume failure.
      20    63   91  26E8  10$:    CMPB    (R3),#^A/ /             ; Less than ^X20 ?
            0E   19  26EB          BLSS    99$                     ; If so, character is not ascii.
   7E 8F    63   91  26ED  20$:    CMPB    (R3),#^A/~/             ; Greater than ^X7E ?
            08   14  26F1          BGTR    99$                     ; If so, character is not ascii.
            53   D6  26F3          INCL    R3                      ; Update pointer.
       F0   52   F5  26F5          SOBGTR  R2,10$                  ; Loop for entire string.
      50    01   D0  26F8          MOVL    #SS$_NORMAL,R0          ; Indicate success.
                 05  26FB  99$:    RSB
                     26FC
                     26FC
```

```
                          26FC  ;++
                          26FC  ;
                          26FC  ; FUNCTIONAL DESCRIPTION:
                          26FC  ;
                          26FC  ;       This routine converts the value in R8 to one ascii format hex digit
                          26FC  ;       and outputs the digit to the format buffer.
                          26FC  ;
                          26FC  ; CALLING SEQUENCE:
                          26FC  ;
                          26FC  ;       JSB     CONVERT_HEX
                          26FC  ;
                          26FC  ; CALLED BY:
                          26FC  ;
                          26FC  ;       LOW_NYBLE
                          26FC  ;       HIGH_NYBLE
                          26FC  ;
                          26FC  ; INPUTS:
                          26FC  ;
                          26FC  ;       R3 -    Points to the position in the format buffer where the result
                          26FC  ;               will be stored.
                          26FC  ;       R8 -    Contains the value to be converted in the low nyble.
                          26FC  ;
                          26FC  ; OUTPUTS:
                          26FC  ;
                          26FC  ;       One ascii character output to the format buffer.
                          26FC  ;
                          26FC  ; IMPLICIT OUTPUTS:
                          26FC  ;
                          26FC  ;       R3 updated.
                          26FC  ;       R8 scratched.
                          26FC  ;
                          26FC  ; RETURN STATUS:
                          26FC  ;
                          26FC  ;       None.
                          26FC  ;
                          26FC  ;--
                          26FC
                          26FC  CONVERT_HEX:
         0A     58    91  26FC          CMPB    R8,#10                  ; Value less than 10?
                05    1E  26FF          BGEQU   10$                     ; Branch if not.
         58     30    80  2701          ADDB    #^X30,R8                ; Convert value 0 through 9 to ascii.
                0A    11  2704          BRB     20$                     ; Jump to common code.
         58     0A    C2  2706  10$:    SUBL    #10,R8                  ; Convert value ^X0A through ^X0F to
58  00000041    8F    C0  2709          ADDL    #^X41,R8                ;  ascii.
         83     58    90  2710  20$:    MOVB    R8,(R3)+                ; Store the converted character.
                      05  2713          RSB
                          2714
                          2714
```

```
                     2714  ;++
                     2714  ;
                     2714  ; FUNCTIONAL DESCRIPTION:
                     2714  ;
                     2714  ;       This routine performs one step of hex to decimal conversion by dividing
                     2714  ;       the quadword in R10 by the quadword in R8.  R8 contains the decimal
                     2714  ;       weight of the digit for this step of the conversion.
                     2714  ;
                     2714  ; CALLING SEQUENCE:
                     2714  ;
                     2714  ;       JSB     CONVERT_DEC
                     2714  ;
                     2714  ; CALLED BY:
                     2714  ;
                     2714  ;       DEC_DISPLAY
                     2714  ;
                     2714  ; INPUTS:
                     2714  ;
                     2714  ;       R8  -   Quadword value for this step from the conversion table.
                     2714  ;       R10 -   Quadword value to be converted.
                     2714  ;
                     2714  ; OUTPUTS:
                     2714  ;
                     2714  ;       R5 -    Decimal digit resulting from this step of the conversion.
                     2714  ;       R8 -    Remainder to be used in next step of conversion.
                     2714  ;
                     2714  ; RETURN STATUS:
                     2714  ;
                     2714  ;       None.
                     2714  ;
                     2714  ;--
                     2714
                     2714  CONVERT_DEC:
        55     94    2714          CLRB    R5                      ; Assume result equals zero.
  5A    58     C2    2716  10$:    SUBL    R8,R10                  ; Quadword subtract with carry.
  5B    59     D9    2719          SBWC    R9,R11                  ;
        04     1F    271C          BCS     20$                     ; Jump if less than zero.
        55     96    271E          INCB    R5                      ; Otherwise result = result + 1.
        F4     11    2720          BRB     10$                     ; Loop.
  5A    58     C0    2722  20$:    ADDL    R8,R10                  ; Quadword add to restore remainder.
  5B    59     D8    2725          ADWC    R9,R11                  ;
               05    2728          RSB
                     2729
                     2729
```

```
2729                .SBTTL  Data display routines.
2729        ;
2729        ; Addresses of data structures are passed to the data display routines in a
2729        ; call frame.  All addresses are determined by the routines CLUSTER_REPORT
2729        ; and LOCAL_REPORT, which determine the format of the report.
2729        ;
2729        ;
2729        ;                  +---------------------------------+
2729        ;        (AP)      |       Number of elements        |
2729        ;                  +---------------------------------+
2729        ; (AP)[CLUB_IN]    |      Base address of CLUB       |
2729        ;                  +---------------------------------+
2729        ;   (AP)[SB_IN]    | Base address of system block    |
2729        ;                  +---------------------------------+
2729        ;  (AP)[CSB_IN]    |       Base address of CSB       |
2729        ;                  +---------------------------------+
2729        ;   (AP)[PB_IN]    |  Base address of path block     |
2729        ;                  +---------------------------------+
2729        ;  (AP)[CDT_IN]    |       Base address of CDT       |
2729        ;                  +---------------------------------+
2729        ;  (AP)[PDT_IN]    |       Base address of PDT       |
2729        ;                  +---------------------------------+
2729        ;   (AP)[MB_IN]    | Address of miscellaneous buffer |
2729        ;                  +---------------------------------+
2729        ;
2729        ;
2729
2729
```

```
                                 2729        .SBTTL  -       GET_DATA routine.
                                 2729  ;++
                                 2729  ;
                                 2729  ; FUNCTIONAL DESCRIPTION:
                                 2729  ;
                                 2729  ;       This routine gets a line of data into the data buffer by calling a
                                 2729  ;       routines that extract and format the data for each of the fields that
                                 2729  ;       are to be displayed.  The data is then stored in DATA_BUF for output.
                                 2729  ;
                                 2729  ; CALLING SEQUENCE:
                                 2729  ;
                                 2729  ;       CALLS   GET_DATA
                                 2729  ;
                                 2729  ; INPUTS:
                                 2729  ;
                                 2729  ;       Addresses of data structures from which the data is to be extracted are
                                 2729  ;       stored in a call frame.  See above picture for details.
                                 2729  ;
                                 2729  ; IMPLICIT INPUTS:
                                 2729  ;
                                 2729  ;       ADRS_TABLE, BITMAP_ADRS, and MAP_LEN must be set up according to the
                                 2729  ;       FIND_ENTRY routine.
                                 2729  ;
                                 2729  ; OUTPUTS:
                                 2729  ;
                                 2729  ;       One line of ascii formatted data stored in DATA_BUF.
                                 2729  ;
                                 2729  ; RETURN STATUS:
                                 2729  ;
                                 2729  ;       R0 =    SS$_NORMAL.
                                 2729  ;
                                 2729  ; REGISTER USAGE:
                                 2729  ;
                                 2729  ;       R3 -    Pointer to next free character position in data buffer.
                                 2729  ;       R6 -    Counts down the number of characters that can be added
                                 2729  ;               to the data buffer before its length exceeds the width
                                 2729  ;               of the screen.
                                 2729  ;       R7 -    Contains the offset into the data structure used to extract
                                 2729  ;               the data.
                                 2729  ;       R10 -   Points to field descriptor.
                                 2729  ;       R11 -   Used to determine the number of fill characters (asterisks,
                                 2729  ;               question marks, or spaces) to add to the display field.
                                 2729  ;
                                 2729  ;--
                                 2729
                           0EC8  2729  .ENTRY  GET_DATA,^M<R3,R6,R7,R9,R10,R11>
        00000476'EF    04    CA  272B          BICL    #MF$M_INVERSE,MISC_FLAGS; Make sure inverse flags starts out
                                 2732                                          ;  clear.
                       00    E1  2732          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
             2C 00000000'EF      2734                  CURRENT_REPORT,3$       ;  cluster report.
          17 00000476'EF    04   E1  273A          BBC     #MF$V_CONT,MISC_FLAGS,2$; Branch if noncontinuous display.
0000225C'EF    00002248'EF   D1  2742          CMPL    LINE_NUM,CMD_LINE       ; Have we overflowed the screen?
                       0A    19  274D          BLSS    2$                      ; Branch if not.
                       18    88  274F          BISB    #SCR$M_MORE_DWN!SCR$M_MORE,-
             0000047A'EF         2751                  SCROLL_FLAGS            ; Indicate screen can be scrolled up.
                     022C    31  2756          BRW     50$                     ; No need to get data for the line.
        00000464'EF          D0  2759  2$:     MOVL    CLSTR_H_SCROLL,-        ; Get a copy of the scroll value for the
```

```
         0000046C'EF      275F               H_SCROLL_LINE              ;  cluster report.
                  0B  11  2764          BRB     4$                      ; Branch to common code.
         00000468'EF  D0  2766  3$:     MOVL    LOCAL_H_SCROLL,-        ; Get a copy of the scroll value for the
         0000046C'EF      276C               H_SCROLL_LINE              ;  local report.
         00000460'EF  D7  2771  4$:     DECL    V_SCROLL_LINE           ; Count down the number of lines of
                          2777                                          ;  data that are scrolled off the
                          2777                                          ;  screen.
                          2777          JGEQ    50$                     ; Skip this line if it's scrolled off
                          277C                                          ;  the screen.
   53    000019A3'EF  DE  277C          MOVAL   DATA_BUF,R3             ; Get pointer to data buffer.
   56    00002222'EF  3C  2783          MOVZWL  SCREEN_WIDTH,R6         ; Get screen width.
                  01  E0  278A          BBS     #SCR$V_MORE_LFT,-       ; Branch if screen has been scrolled
      08 0000047A'EF      278C               SCROLL_FLAGS,5$            ;  left.
                FC8A  30  2792          BSBW    DELM                    ; Add starting field delimiter to line.
                          2795          JLEQU   40$                     ; Branch if it won't fit on the screen.
         00002270'EF  D4  279A  5$:     CLRL    START_BIT               ; Start scanning the bitmap with bit
                          27A0                                          ;  zero.
                  00  E1  27A0          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
      09 00000000'EF      27A2               CURRENT_REPORT,6$          ;  cluster report.
   0000226C'EF    02  D0  27A8          MOVL    #CLUSTER_MAP_LEN,MAP_LEN; Save length of cluster bitmap.
                  07  11  27AF          BRB     10$                     ; Jump to common code.
   0000226C'EF    00  D0  27B1  6$:     MOVL    #LOCAL_MAP_LEN,MAP_LEN  ; Save length of local bitmap.
                FBA8  30  27B8  10$:    BSBW    FIND_ENTRY              ; Find an entry.
                          27BB          JLBC    R0,40$                  ; Branch if an entry wasn't found.
               0C AA  D3  27C1          BITL    FD$L_CLASS(R10),-       ; Is this entry in a different class?
         00000001'EF      27C4               CLASS_SELECT               ;
                  ED  13  27C9          BEQLU   10$                     ; Branch if so.
               0C AA  D3  27CB          BITL    FD$L_CLASS(R10),-       ; Is this class enabled?
         000021F8'EF      27CE               DATA_CNTRL                 ;
                  11  12  27D3          BNEQU   20$                     ; Branch if so.
                          27D5  ;
                          27D5  ; Class is disabled, so fill this field with spaces.
                          27D5  ;
                FC6B  30  27D5          BSBW    ADD_ONE_SPACE           ; Preceed the data with atleast one
                          27D8                                          ;  space.
                          27D8          JLEQU   40$                     ; Branch if it won't fit.
          59      6A  9A  27DD          MOVZBL  FD$B_FIELD_SIZE(R10),R9 ; Get the size of the field.
                FC66  30  27E0          BSBW    ADD_SPACES              ; Fill the field with spaces.
                0106  31  27E3          BRW     29$                     ; Jump to common code.
                          27E6
          57   10 AA  D0  27E6  20$:    MOVL    FD$L_OFFSET(R10),R7     ; Get the SCS offset.
               08 BA  16  27EA          JSB     @FD$L_DISPLAY_ROUTINE(R10)
                          27ED                                          ; Call the display routine to extract
                          27ED                                          ;  and format the data.
          59      50  D0  27ED          MOVL    R0,R9                   ; Save status from display routine.
                  02  E1  27F0          BBC     #MF$V_INVERSE,-         ; Branch if the data hasn't changed
      20 00000476'EF      27F2               MISC_FLAGS,22$             ;  since the last snapshot.
00 00000434'EF   00002248'EF  E2  27F8  BBSS    LINE_NUM,LINE_DIF,21$   ; Indicate the some data in this line
                          2804                                          ;  changed and the line must be
                          2804                                          ;  rewritten to the screen.
   0C 00002247'EF     01  E1  2804  21$:    BBC     #DVT$V_DCRT,OUT_DEV,22$ ; Branch if not DECcrt terminal.
         0000017E'EF  28  280C          MOVC3   INVRS_DSCR,-            ; Store sequence to turn on inverse
   63    00000186'EF      2812               INVRS_DSCR+8,(R3)          ;  video.
                FC28  30  2818  22$:    BSBW    ADD_ONE_SPACE           ; Preceed data with one space.
                          281B                                          ; Branch to display subroutine for this
                          281B                                          ;  field.
                          281B          JLEQU   40$                     ; Branch if it didn't fit.
```

```
                  2E 59   E8  2820              BLBS    R9,25$                  ; Branch if display routine succeeded.
                              2823      ;
                              2823      ; Display routine failed, so center a question mark in the field.
                              2823      ;
              59    6A    9A  2823              MOVZBL  FD$B_FIELD_SIZE(R10),R9 ; Get the size of this field.
                    59    97  2826              DECB    R9                      ; Subtract one for the question mark.
      59    59   FF 8F    9C  2828              ROTL    #-1,R9,R9               ; Calculate the number of spaces needed
                              282D                                              ;  to center the question mark.
                  FC19    30  282D              BSBW    ADD_SPACES              ; Add leading spaces to buffer.
                              2830              JLEQU   40$                     ; Branch if the spaces won't fit on the
                              2835                                              ;  screen.
              08 59   1F  E1  2835              BBC     #31,R9,23$              ; Branch if even number of spaces.
                  FC07    30  2839              BSBW    ADD_ONE_SPACE           ; Add one more leading space.
                              283C              JLEQU   40$                     ; Branch if it didn't fit.
              83    3F    90  2841      23$:    MOVB    #^A/?/,(R3)+            ; Store the question mark.
                    56    D7  2844              DECL    R6                      ; Update number of columns remaining.
                              2846              JLEQU   40$                     ; Branch if the question mark didn't
                              284B                                              ;  fit on the screen.
                  FBFB    30  284B              BSBW    ADD_SPACES              ; Add trailing spaces to buffer.
                  009B    31  284E              BRW     29$                     ; Jump to common code.
                              2851
              59    6A    9A  2851      25$:    MOVZBL  FD$B_FIELD_SIZE(R10),R9 ; Get the size of this field.
      59    00002153'EF   D1  2854              CMPL    FORMAT_BUF_LEN,R9       ; Will the formatted data fit in the
                              285B                                              ;  display field?
                    78    13  285B              BEQLU   28$                     ; Branch if it fits exactly.
                    24    1F  285D              BLSSU   27$                     ; Branch if its smaller.
                 05 AA    91  285F              CMPB    FD$B_FORMAT(R10),-      ; Is the data formatted in hex?
                    02        2862                      #HEX                    ;
                    0F    13  2863              BEQLU   26$                     ; Branch if so.
                 05 AA    91  2865              CMPB    FD$B_FORMAT(R10),-      ; Is the data formatted in decimal?
                    03        2868                      #DEC                    ;
                    09    13  2869              BEQLU   26$                     ; Branch if so.
                              286B      ;
                              286B      ; The data is ascii, but is larger than the field, so truncate the string.
                              286B      ;
                              286B
      00002153'EF    59   D0  286B              MOVL    R9,FORMAT_BUF_LEN       ; Truncate the string by making its
                              2872                                              ;  length equal to the field size.
                    61    11  2872              BRB     28$                     ; Copy the string.
                              2874
                              2874      ;
                              2874      ; The data is non ascii and is larger than the field, so fill the field with
                              2874      ; asterisks.
                              2874      ;
                              2874
                    34    BB  2874      26$:    PUSHR   #^M<R2,R4,R5>
   63    59    2A   63    00  2C  2876          MOVC5   #0,(R3),#^A/*/,R9,(R3)  ; Fill field with asterisks.
                    34    BA  287C              POPR    #^M<R2,R4,R5>
              56    59    C2  287E              SUBL    R9,R6                   ; See if it fit on the screen.
                    69    11  2881              BRB     29$                     ; Jump to common code.
                              2883      ;
                              2883      ; If the string is ascii then left justify the string in the field.  If
                              2883      ; hex or decimal then right justify the string.
                              2883      ;
                              2883
                              2883      27$:
      59    00002153'EF   C2  2883              SUBL    FORMAT_BUF_LEN,R9       ; Get number of spaces to add.
```

```
                  02     05 AA   91  288A          CMPB    FD$B_FORMAT(R10),#HEX   ; Is the data to be displayed in
                                     288E                                          ;  hexadecimal?
                         24      13  288E          BEQL    275$                    ; Branch if so.
                  03     05 AA   91  2890          CMPB    FD$B_FORMAT(R10),#DEC   ; Is the data to be displayed in
                                     2894                                          ;  decimal?
                         1E      13  2894          BEQL    275$                    ; Branch if so.
                         34      BB  2896          PUSHR   #^M<R2,R4,R5>
               00002153'EF       28  2898          MOVC3   FORMAT_BUF_LEN,-        ; Copy the string into the data buffer.
          63   0000215B'EF           289E                  FORMAT_BUF,(R3)         ;
                         34      BA  28A4          POPR    #^M<R2,R4,R5>
          56   00002153'EF       C2  28A6          SUBL    FORMAT_BUF_LEN,R6       ; Calculate columns remaining.
                         68      1B  28AD          BLEQU   40$                     ; Branch if the string didn't fit on
                                     28AF                                          ;  the screen.
                       FB97      30  28AF          BSBW    ADD_SPACES              ; Add trailing spaces.
                         38      11  28B2          BRB     29$                     ; Branch to common code.
                                     28B4  275$:
                       FB92      30  28B4          BSBW    ADD_SPACES              ; Add leading spaces.
                                     28B7          JLEQU   40$                     ; Branch if the spaces didn't fit on
                                     28BC                                          ;  the screen.
                         34      BB  28BC          PUSHR   #^M<R2,R4,R5>
               00002153'EF       28  28BE          MOVC3   FORMAT_BUF_LEN,-        ; Copy the string into the data buffer.
          63   0000215B'EF           28C4                  FORMAT_BUF,(R3)         ;
                         34      BA  28CA          POPR    #^M<R2,R4,R5>
          56   00002153'EF       C2  28CC          SUBL    FORMAT_BUF_LEN,R6       ; Calculate columns remaining.
                         17      11  28D3          BRB     29$                     ; Branch to common code.
                         34      BB  28D5  28$:    PUSHR   #^M<R2,R4,R5>
               00002153'EF       28  28D7          MOVC3   FORMAT_BUF_LEN,-        ; Copy the string into the data buffer.
          63   0000215B'EF           28DD                  FORMAT_BUF,(R3)         ;
                         34      BA  28E3          POPR    #^M<R2,R4,R5>
          56   00002153'EF       C2  28E5          SUBL    FORMAT_BUF_LEN,R6       ; Will it fit on the screen?
                         29      1B  28EC  29$:    BLEQU   40$                     ; Branch if it didn't fit.
                       FB52      30  28EE          BSBW    ADD_ONE_SPACE           ; Follow with one space.
                         24      1B  28F1          BLEQU   40$                     ; Branch if it didn't fit.
                         02      E5  28F3          BBCC    #MF$V_INVERSE,-         ; Branch if the data hasn't changed
             14 00000476'EF          28F5                  MISC_FLAGS,30$          ;  since the last snapshot.
          0C 00002247'EF    01   E1  28FB          BBC     #DVT$V_DCRT,OUT_DEV,30$ ; Branch if not DECcrt terminal.
               00000143'EF       28  2903          MOVC3   DCRT_NOATR_DSCR,-       ; Store sequence to turn off attributes
          63   0000014B'EF           2909                  DCRT_NOATR_DSCR+8,(R3)  ;  (inverse video).
                       FB0D      30  290F  30$:    BSBW    DELM                    ; Add ending delimiter to line.
                                     2912          JGTRU   10$                     ; Loop if there's more room on the line.
                         02      E5  2917  40$:    BBCC    #MF$V_INVERSE,-         ; Branch if the data hasn't changed
             14 00000476'EF          2919                  MISC_FLAGS,45$          ;  since the last snapshot.
          0C 00002247'EF    01   E1  291F          BBC     #DVT$V_DCRT,OUT_DEV,45$ ; Branch if not DECcrt terminal.
               00000143'EF       28  2927          MOVC3   DCRT_NOATR_DSCR,-       ; Store sequence to turn off attributes
          63   0000014B'EF           292D                  DCRT_NOATR_DSCR+8,(R3)  ;  (inverse video).
0000199B'EF    53 000019A3'8F    C3  2933  45$:    SUBL3   #DATA_BUF,R3,DATA_DSCR  ; Save length of data line.
               5B 0000199B'EF    DE  293F          MOVAL   DATA_DSCR,R11           ; Get address of descriptor for the
                                     2946                                          ;  line of data.
                                     2946
          34 00000476'EF    03   E0  2946          BBS     #MF$V_RIP,MISC_FLAGS,49$; If refresh in progress, make sure
                                     294E                                          ;  we draw all the lines.
   28 0000043C'EF   00002248'EF  E0  294E          BBS     LINE_NUM,BLANK_LINES,49$; Branch if this is a new blank line.
   1C 00000434'EF   00002248'EF  E0  295A          BBS     LINE_NUM,LINE_DIF,49$   ; Branch if this line has changed.
          0C 00002247'EF    01   E1  2966          BBC     #DVT$V_DCRT,OUT_DEV,48$ ; Branch if output device is not a
                                     296E                                          ;  DECcrt terminal.
               00002248'EF       E0  296E          BBS     LINE_NUM,-              ; Branch if this line changed on the
             08 00000438'EF          2974                  PRIOR_LINE_DIF,49$      ;  last pass and must be rewritten to
```

```
                          297A  7775                                          ;  remove the inverse video.
         00002248'EF  D6  297A  7776 48$:     INCL    LINE_NUM                ; Otherwise, just bump the line number.
                  03  11  2980  7777          BRB     50$                     ; And finish up.
                FAE1  30  2982  7778 49$:     BSBW    DRAW_LINE               ; Output the line.
            50    01  D0  2985  7779 50$:     MOVL    #SS$_NORMAL,R0
                      04  2988  7780          RET
                          2989  7781
                          2989  7782
```

```
                 2989          .SBTTL  -       General data display routines.
                 2989  ;++
                 2989  ;
                 2989  ; FUNCTIONAL DESCRIPTION:
                 2989  ;
                 2989  ;       This is the default data display routine.  It gets the address of the
                 2989  ;       data by first determining the type of structure containing the data
                 2989  ;       by examining the prefix of the structure name argument.  If the prefix
                 2989  ;       is valid, it is converted to an index into the call frame which
                 2989  ;       contains the base address of the data structures containing the data.
                 2989  ;       the base address of the proper structure is then added to R7 to get
                 2989  ;       the address of the data. FORMAT_DATA is then called to format the data.
                 2989  ;
                 2989  ; Note: DISPLAY_FIELD can only be used when the data to be displayed is found
                 2989  ;       in either the club, a cluster system block, an SCS system block, path
                 2989  ;       block, or connection descriptor table, a port descriptor table, or the
                 2989  ;       miscellaneous buffer.  The argument STRUCTURE_NAME given in the
                 2989  ;       DEFINE_FIELD macro must be a valid SCS offset and must begin with CLUB.
                 2989  ;       SB$, PB$, CDT$, PDT$ or a miscellaneous buffer offset begining with MB$.
                 2989  ;       Any other argument will cause the field to be filled with question
                 2989  ;       marks.
                 2989  ;
                 2989  ;
                 2989  ; CALLING SEQUENCE:
                 2989  ;
                 2989  ;       JSB     DISPLAY_FIELD
                 2989  ;
                 2989  ; CALLED BY:
                 2989  ;
                 2989  ;       GET_DATA
                 2989  ;
                 2989  ; INPUTS:
                 2989  ;
                 2989  ;       R7      = SCS offset from DEFINE_FIELD macro.
                 2989  ;       R10     = Address of field descriptor.
                 2989  ;
                 2989  ;
                 2989  ; IMPLICIT INPUTS:
                 2989  ;
                 2989  ;       AP      = Address of call frame containing the addresses of the data
                 2989  ;                 structures to be used when data is extracted.
                 2989  ;
                 2989  ; OUTPUTS:
                 2989  ;
                 2989  ;       Formated data string returned in FORMAT_BUF.
                 2989  ;       Length of formatted data string returned in FORMAT_BUF_LEN.
                 2989  ;
                 2989  ;
                 2989  ; RETURN STATUS:
                 2989  ;
                 2989  ;       R0 =    SS$_NORMAL or failed, or status from FORMAT_DATA.
                 2989  ;
                 2989  ;--
                 2989
                 2989  DISPLAY_FIELD:
   01FC 8F   BB  2989          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>
        50   D4  298D          CLRL    R0                      ; Assume failure.
```

```
           58      01  D0  298F  7841          MOVL    #1,R8                     ; Set index past number of arguments in
                           2992  7842                                            ;  call frame.
  54    00000489'EF    DE  2992  7843          MOVAL   SYM_PRFX,R4               ; Point to array of symbol prefixes.
        55      14 AA  DE  2999  7844          MOVAL   FD$[_OFFSET_PRFX(R10),R5; Get the address of the src2 string.
           53      84  9A  299D  7845 10$:     MOVZBL  (R4)+,R3                  ; Get the length of src1 string
                   08  BB  29A0  7846          PUSHR   #^M<R3>
      65   64      53  29  29A2  7847          CMPC    R3,(R4),(R5)              ; Check for valid prefix.
                   08  BA  29A6  7848          POPR    #^M<R3>
                   0B  13  29A8  7849          BEQL    20$                       ; Branch if valid prefix.
           54      53  C0  29AA  7850          ADDL    R3,R4                     ; Point to next prefix.
  FFEA 58  01      56  F1  29AD  7851          ACBL    R6,#1,R8,10$              ; Loop for all arguments.
                   2C  11  29B3  7852          BRB     99$                       ; Branch if invalid prefix.
  54    0000049C'8F    D1  29B5  7853 20$:     CMPL    #CDT_PFX+1,R4             ; Is the prefix "CDT$"?
                   09  12  29BC  7854          BNEQ    30$                       ; Branch if not.
  57    FFFFFFF0 8F    C0  29BE  7855          ADDL    #SHC$L_CDT_DIF,R7         ; Adjust CDT offset to compensate for
                           29C5  7856                                            ;  portion of CDT not stored.
                   10  11  29C5  7857          BRB     40$                       ; Skip check for "PDT$".
  54    000004A1'8F    D1  29C7  7858 30$:     CMPL    #PDT_PFX+1,R4             ; Is the prefix "PDT$"?
                   07  12  29CE  7859          BNEQ    40$                       ; Branch if not.
  57    FFFFFF4C 8F    C0  29D0  7860          ADDL    #SHC$L_PDT_DIF,R7         ; Adjust PDT offset to compensate for
                           29D7  7861                                            ;  portion of PDT not stored.
        57       6C48  C0  29D7  7862 40$:     ADDL    (R12)[R8],R7              ; Get address of data.
                 035F  30  29DB  7863          BSBW    CHECK_DIF                 ; Check for change in data since last
                           29DE  7864                                            ;  scan.
                 FB9E  30  29DE  7865          BSBW    FORMAT_DATA               ; Format the data.
             01FC 8F   BA  29E1  7866 99$:     POPR    #^M<R2,R3,R4,R5,R6,R7,R8>
                       05  29E5  7867          RSB
                           29E6  7868
                           29E6  7869
```

```
                    29E6  ;++
                    29E6  ;
                    29E6  ; FUNCTIONAL DESCRIPTION:
                    29E6  ;
                    29E6  ;       This routine is used to convert a numeric value into a string.  The
                    29E6  ;       value is used as an index into a list of counted ascii strings.  The
                    29E6  ;       address of the indexed string is then used when calling the FORMAT_DATA
                    29E6  ;       routine.
                    29E6  ;
                    29E6  ; Note: The FORMAT argument in the field descriptor for any field using this
                    29E6  ;       routine must be of the type "ASCIC".
                    29E6  ;
                    29E6  ; CALLING SEQUENCE:
                    29E6  ;
                    29E6  ;       JSB     DISPLAY_STRING
                    29E6  ;
                    29E6  ; CALLED BY:
                    29E6  ;
                    29E6  ;       DISPLAY_RPORT_TYPE
                    29E6  ;       DISPLAY_CABLE_STATUS
                    29E6  ;       DISPLAY_LOOPBACK_STATUS
                    29E6  ;
                    29E6  ; INPUTS:
                    29E6  ;
                    29E6  ;       R7      = Value to be used as an index into a list of counted ascii
                    29E6  ;                 strings.
                    29E6  ;       R8      = Address of a list of counted ascii strings.
                    29E6  ;
                    29E6  ; OUTPUTS:
                    29E6  ;
                    29E6  ;       Formated data string returned in FORMAT_BUF.
                    29E6  ;       Length of formatted data string returned in FORMAT_BUF_LEN.
                    29E6  ;
                    29E6  ;
                    29E6  ; RETURN STATUS:
                    29E6  ;
                    29E6  ;       R0 =    SS$_NORMAL or failed, or status from FORMAT_DATA.
                    29E6  ;
                    29E6  ;--
                    29E6
                    29E6  DISPLAY_STRING:
0380 8F       BB    29E6          PUSHR   #^M<R7,R8,R9>
     57       D5    29EA          TSTL    R7                      ; Index equal zero?
     09       13    29EC          BEQLU   20$                     ; Branch if so.
59   88       9A    29EE  10$:    MOVZBL  (R8)+,R9                ; Get the number of characters in the
                    29F1                                          ;  descriptor.
58   59       C0    29F1          ADDL    R9,R8                   ; Point to next descriptor.
  F7 57       F5    29F4          SOBGTR  R7,10$                  ; Loop till R7 equals 0 and R8 points
                    29F7                                          ;  to the string.
57   58       C0    29F7  20$:    ADDL    R8,R7                   ; Get address of string.
   FB82       30    29FA          BSBW    FORMAT_DATA             ; Format the string.
0380 8F       BA    29FD          POPR    #^M<R7,R8,R9>
              05    2A01          RSB
                    2A02
                    2A02
```

```
                        2A02          .SBTTL  -       Cluster class display routines.
                        2A02
                        2A02   ;++
                        2A02   ;
                        2A02   ; FUNCTIONAL DESCRIPTION:
                        2A02   ;
                        2A02   ;       The following routines are used to display the bit in the CLUB$L_FLAGS
                        2A02   ;       longword in the CLUB and the CSB$L_STATUS longword in the CSB.  These
                        2A02   ;       bits are displayed as a one or zero only.
                        2A02   ;
                        2A02   ; CALLING SEQUENCE:
                        2A02   ;
                        2A02   ;       JSB     DISPLAY_QF_VOTE
                        2A02   ;       JSB     DISPLAY_QF_SAME
                        2A02   ;       JSB     DISPLAY_QF_ACTIVE
                        2A02   ;
                        2A02   ; CALLED BY:
                        2A02   ;
                        2A02   ;       GET_DATA
                        2A02   ;
                        2A02   ; INPUTS:
                        2A02   ;
                        2A02   ;       R7      = Offset of longword containing the bit to display.
                        2A02   ;       (AP)[CLUB_IN]   = Base address of club or cluster system block.
                        2A02   ;
                        2A02   ; IMPLICIT INPUTS:
                        2A02   ;
                        2A02   ;       AP      = Address of call frame containing the addresses of the data
                        2A02   ;                 structures to be used when data is extracted.
                        2A02   ;
                        2A02   ; OUTPUTS:
                        2A02   ;
                        2A02   ;       Formated data string returned in FORMAT_BUF.
                        2A02   ;       Length of formatted data string returned in FORMAT_BUF_LEN.
                        2A02   ;
                        2A02   ; REGISTER USAGE:
                        2A02   ;
                        2A02   ;       R7      = Offset into data structure where data is located.  Converted
                        2A02   ;                 into actual address of data.
                        2A02   ;       R8      = Index into call frame to locate structure base address.
                        2A02   ;--
                        2A02
                        2A02   DISPLAY_QF_VOTE:
  01C0 8F       BB      2A02          PUSHR   #^M<R6,R7,R8>
    58 01       D0      2A06          MOVL    #CLUB_IN,R8             ; Set index into call frame to club
                        2A09                                          ;  base address.
  57  6C48      C0      2A09          ADDL    (R12)[R8],R7            ; Get address of data.
   56   19      D0      2A0D          MOVL    #CLUB$V_QF_VOTE,R6      ; Get number of bit to check.
      0031      30      2A10          BSBW    DISPLAY_QF              ; Convert to formatted string.
  01C0 8F       BA      2A13          POPR    #^M<R6,R7,R8>
        05              2A17          RSB
                        2A18
                        2A18   DISPLAY_QF_SAME:
  01C0 8F       BB      2A18          PUSHR   #^M<R6,R7,R8>
    58 03       D0      2A1C          MOVL    #CSB_IN,R8              ; Set index into call frame to cluster
                        2A1F                                          ;  system block base address.
  57  6C48      C0      2A1F          ADDL    (R12)[R8],R7            ; Get address of data.
```

```
            56    03  D0  2A23  7984              MOVL    #CSB$V_QF_SAME,R6       ; Get number of bit to check.
                001B  30  2A26  7985              BSBW    DISPLAY_QF              ; Convert to formatted string.
          01C0 8F     BA  2A29  7986              POPR    #^M<R6,R7,R8>
                      05  2A2D  7987              RSB
                          2A2E  7988
                          2A2E  7989 DISPLAY_QF_ACTIVE:
          01C0 8F     BB  2A2E  7990              PUSHR   #^M<R6,R7,R8>
            58    03  D0  2A32  7991              MOVL    #CSB_IN,R8              ; Set index into call frame to cluster
                          2A35  7992                                              ;  system block base address.
          57    6C48  C0  2A35  7993              ADDL    (R12)[R8],R7            ; Get address of data.
            56    09  D0  2A39  7994              MOVL    #CSB$V_QF_ACTIVE,R6     ; Get number of bit to check.
                0005  30  2A3C  7995              BSBW    DISPLAY_QF              ; Convert to formatted string.
          01C0 8F     BA  2A3F  7996              POPR    #^M<R6,R7,R8>
                      05  2A43  7997              RSB
                          2A44  7998
                          2A44  7999 ;
                          2A44  8000 ; Convert the bit into an ascii string "YES" or "NO".
                          2A44  8001 ;
                          2A44  8002
                          2A44  8003 DISPLAY_QF:
                02F6  30  2A44  8004              BSBW    CHECK_DIF               ; Check for change in data since last
                          2A47  8005                                              ;  update.
       04 67      56  E1  2A47  8006              BBC     R6,(R7),10$             ; Branch if QF_VOTE isn't set.
                  57  D4  2A4B  8007              CLRL    R7                      ; Get index into string array.
                  03  11  2A4D  8008              BRB     20$                     ; Branch to common.
            57    01  D0  2A4F  8009 10$:         MOVL    #1,R7                   ; Get index into string array.
    58  00000404'EF   DE  2A52  8010 20$:         MOVAL   YES_NO_STRING,R8        ; Get address of string array.
                FF8A  30  2A59  8011              BSBW    DISPLAY_STRING          ; Format the string.
                      05  2A5C  8012              RSB
                          2A5D  8013
```

```
                            2A5D          .SBTTL  -       Systems class display routines.
                            2A5D
                            2A5D    ;++
                            2A5D    ;
                            2A5D    ; FUNCTIONAL DESCRIPTION:
                            2A5D    ;
                            2A5D    ;         This routine is used to combine SB$T_SWTYPE and SB$T_SWVERS to
                            2A5D    ;         produce the SOFTWARE field.
                            2A5D    ;
                            2A5D    ; CALLING SEQUENCE:
                            2A5D    ;
                            2A5D    ;         JSB     DISPLAY_SOFTWARE
                            2A5D    ;
                            2A5D    ; CALLED BY:
                            2A5D    ;
                            2A5D    ;         GET_DATA
                            2A5D    ;
                            2A5D    ; INPUTS:
                            2A5D    ;
                            2A5D    ;         R7      = Offset to SB$T_SWVERS in the system block.
                            2A5D    ;         (AP)[SB_IN]     = Address of system block.
                            2A5D    ;
                            2A5D    ; IMPLICIT INPUTS:
                            2A5D    ;
                            2A5D    ;         AP      = Address of call frame containing the addresses of the data
                            2A5D    ;                   structures to be used when data is extracted.
                            2A5D    ;
                            2A5D    ; OUTPUTS:
                            2A5D    ;
                            2A5D    ;         Formated data string returned in FORMAT_BUF.
                            2A5D    ;         Length of formatted data string returned in FORMAT_BUF_LEN.
                            2A5D    ;
                            2A5D    ; RETURN STATUS:
                            2A5D    ;
                            2A5D    ;         Status returned from FORMAT_DATA routine.
                            2A5D    ;
                            2A5D    ; REGISTER USAGE:
                            2A5D    ;
                            2A5D    ;         R7      = Offset into data structure where data is located.  Converted
                            2A5D    ;                   into actual address of data.
                            2A5D    ;         R8      = Index into call frame to locate system block base address.
                            2A5D    ;--
                            2A5D
                            2A5D    DISPLAY_SOFTWARE:
           0300 8F     BB   2A5D            PUSHR   #^M<R8,R9>
               FF25    30   2A61            BSBW    DISPLAY_FIELD           ; Format the software version.
 59   0000215B'EF      D0   2A64            MOVL    FORMAT_BUF,R9           ; Save the version temporarily.
          58    02     D0   2A6B            MOVL    #SB_IN,R8               ; Set index into call frame to system
                            2A6E                                            ;  block base address.
          57    24     D0   2A6E            MOVL    #SB$T_SWTYPE,R7         ; Get software type offset.
          57  6C48     C0   2A71            ADDL    (R12)[R8],R7            ; Get address of data.
   0000215B'EF  67     D0   2A75            MOVL    (R7),FORMAT_BUF         ; Store software type in buffer.
   0000215E'EF  20     90   2A7C            MOVB    #^A/ /,FORMAT_BUF+3     ; Place a space between type and
                            2A83                                            ;  version.
   0000215F'EF  59     D0   2A83            MOVL    R9,FORMAT_BUF+4         ; Store the version in the buffer.
   00002153'EF  08     D0   2A8A            MOVL    #8,FORMAT_BUF_LEN       ; Buffer length = 8 characters.
           0300 8F     BA   2A91            POPR    #^M<R8,R9>
```

```
05  2A95  8072          RSB
    2A96  8073
    2A96  8074
```

```
                         2A96      .SBTTL  -       Cluster system block display routines.
                         2A96
                         2A96 ;++
                         2A96 ;
                         2A96 ; FUNCTIONAL DESCRIPTION:
                         2A96 ;
                         2A96 ;        This routine is used to display the protocol version and ECO level.
                         2A96 ;
                         2A96 ; CALLING SEQUENCE:
                         2A96 ;
                         2A96 ;        JSB     DISPLAY_PROTOCOL
                         2A96 ;
                         2A96 ; CALLED BY:
                         2A96 ;
                         2A96 ;        GET_DATA
                         2A96 ;
                         2A96 ; INPUTS:
                         2A96 ;
                         2A96 ;        R7      = SCS offset CSB$B_ECOLVL from DEFINE_FIELD macro.
                         2A96 ;        (AP)[CSB_IN]    = Base address of selected cluster system block.
                         2A96 ;
                         2A96 ; IMPLICIT INPUTS:
                         2A96 ;
                         2A96 ;        AP      = Address of call frame containing the addresses of the data
                         2A96 ;                  structures to be used when data is extracted.
                         2A96 ;
                         2A96 ; OUTPUTS:
                         2A96 ;
                         2A96 ;        Formated ascii string returned in FORMAT_BUF.
                         2A96 ;        Length of formatted ascii string returned in FORMAT_BUF_LEN.
                         2A96 ;
                         2A96 ; RETURN STATUS:
                         2A96 ;
                         2A96 ;        R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                         2A96 ;
                         2A96 ;--
                         2A96
                         2A96 DISPLAY_PROTOCOL:
         0180 8F    BB   2A96          PUSHR   #^M<R7,R8>
           58  03   D0   2A9A          MOVL    #CSB_IN,R8              ; Set index into call frame to path
                         2A9D                                          ;  block base address.
         57  6C48   C0   2A9D          ADDL    (R12)[R8],R7            ; Get the address of the data.
             0299   30   2AA1          BSBW    CHECK_DIF               ; Check for change in data since last
                         2AA4                                          ;  scan.
00000018'EF  67  08 78   2AA4          ASHL    #8,(R7),SCRATCH         ; Store eco level in 2nd byte of
                         2AAC                                          ;  scratch location.
   57  00000041 8F  D0   2AAC          MOVL    #CSB$B_VERNUM,R7        ; Get version number offset.
         57  6C48   C0   2AB3          ADDL    (R12)[R8],R7            ; Get the address of the data.
             0283   30   2AB7          BSBW    CHECK_DIF               ; Check for change in data since last
                         2ABA                                          ;  scan.
   00000018'EF  67  90   2ABA          MOVB    (R7),SCRATCH            ; Store version number in 1st byte of
                         2AC1                                          ;  scratch location.
   57  00000018'EF  DE   2AC1          MOVAL   SCRATCH,R7              ; Get address of data.
             FAB4   30   2AC8          BSBW    FORMAT_DATA             ; Format the data.
         0180 8F    BA   2ACB          POPR    #^M<R7,R8>
               05        2ACF          RSB
                         2AD0
```

```
2AD0  8133
```

```
                          2AD0  8135
                          2AD0  8136 ;++
                          2AD0  8137 ;
                          2AD0  8138 ; FUNCTIONAL DESCRIPTION:
                          2AD0  8139 ;
                          2AD0  8140 ;       This routine is used to display the state of the cluster system.
                          2AD0  8141 ;
                          2AD0  8142 ; CALLING SEQUENCE:
                          2AD0  8143 ;
                          2AD0  8144 ;       JSB     DISPLAY_CSB_STATE
                          2AD0  8145 ;
                          2AD0  8146 ; CALLED BY:
                          2AD0  8147 ;
                          2AD0  8148 ;       GET_DATA
                          2AD0  8149 ;
                          2AD0  8150 ; INPUTS:
                          2AD0  8151 ;
                          2AD0  8152 ;       R7              = SCS offset CSB$B_STATE from DEFINE_FIELD macro.
                          2AD0  8153 ;       (AP)[CSB_IN]    = Base address of selected cluster system block.
                          2AD0  8154 ;
                          2AD0  8155 ; IMPLICIT INPUTS:
                          2AD0  8156 ;
                          2AD0  8157 ;       AP      = Address of call frame containing the addresses of the data
                          2AD0  8158 ;                 structures to be used when data is extracted.
                          2AD0  8159 ;
                          2AD0  8160 ; OUTPUTS:
                          2AD0  8161 ;
                          2AD0  8162 ;       Formated ascii string returned in FORMAT_BUF.
                          2AD0  8163 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                          2AD0  8164 ;
                          2AD0  8165 ; RETURN STATUS:
                          2AD0  8166 ;
                          2AD0  8167 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                          2AD0  8168 ;
                          2AD0  8169 ;--
                          2AD0  8170
                          2AD0  8171 DISPLAY_CSB_STATE:
        0180 8F     BB    2AD0  8172         PUSHR   #^M<R7,R8>
        58    03    D0    2AD4  8173         MOVL    #CSB_IN,R8              ; Set index into call frame to path
                          2AD7  8174                                         ;  block base address.
     57     6C48    C0    2AD7  8175         ADDL    (R12)[R8],R7            ; Get the address of the data.
            025F    30    2ADB  8176         BSBW    CHECK_DIF               ; Check for change in data since last
                          2ADE  8177                                         ;  scan.
 58    0000040B'EF  DE    2ADE  8178         MOVAL   CSB_STATE_STRNG,R8      ; Point to array of state descriptors.
        57    67    9A    2AE5  8179         MOVZBL  (R7),R7                 ; Get the state.
              57    D7    2AE8  8180         DECL    R7                      ; Subtract one so index starts at zero.
            FEF9    30    2AEA  8181         BSBW    DISPLAY_STRING          ; Format the string.
        0180 8F     BA    2AED  8182         POPR    #^M<R7,R8>
              05          2AF1  8183         RSB
                          2AF2  8184
                          2AF2  8185
```

```
                              2AF2
                              2AF2    ;++
                              2AF2    ;
                              2AF2    ; FUNCTIONAL DESCRIPTION:
                              2AF2    ;
                              2AF2    ;         This routine is used to display the state of the cluster system.
                              2AF2    ;
                              2AF2    ;
                              2AF2    ; CALLING SEQUENCE:
                              2AF2    ;
                              2AF2    ;         JSB     DISPLAY_CSB_STATUS
                              2AF2    ;
                              2AF2    ; CALLED BY:
                              2AF2    ;
                              2AF2    ;         GET_DATA
                              2AF2    ;
                              2AF2    ; INPUTS:
                              2AF2    ;
                              2AF2    ;         R7      = SCS offset CSB$L_STATUS from DEFINE_FIELD macro.
                              2AF2    ;         (AP)[CSB_IN]    = Base address of selected cluster system block.
                              2AF2    ;
                              2AF2    ; IMPLICIT INPUTS:
                              2AF2    ;
                              2AF2    ;         AP      = Address of call frame containing the addresses of the data
                              2AF2    ;                   structures to be used when data is extracted.
                              2AF2    ;
                              2AF2    ; OUTPUTS:
                              2AF2    ;
                              2AF2    ;         Formated ascii string returned in FORMAT_BUF.
                              2AF2    ;         Length of formatted ascii string returned in FORMAT_BUF_LEN.
                              2AF2    ;
                              2AF2    ; RETURN STATUS:
                              2AF2    ;
                              2AF2    ;         R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                              2AF2    ;
                              2AF2    ;--
                              2AF2
                              2AF2    ASSUME CSB$V_LONG_BREAK EQ 0
                              2AF2    ASSUME CSB$V_MEMBER     EQ 1
                              2AF2    ASSUME CSB$V_REMOVED    EQ 2
                              2AF2
                              2AF2    DISPLAY_CSB_STATUS:
           0180 8F       BB   2AF2            PUSHR   #^M<R7,R8>
           58   03       D0   2AF6            MOVL    #CSB_IN,R8              ; Set index into call frame to path
                              2AF9                                            ;  block base address.
         57   6C48       C0   2AF9            ADDL    (R12)[R8],R7            ; Get the address of the data.
              023D       30   2AFD            BSBW    CHECK_DIF               ; Check for change in data since last
                              2B00                                            ;  scan.
           57   67       90   2B00            MOVB    (R7),R7                 ; Get the status.
57   FFFFFFF8 8F         CA   2B03            BICL    #^C<CSB$M_LONG_BREAK!-  ; Clear any non-status bits.
                              2B0A                    CSB$M_MEMBER!-          ;
                              2B0A                    CSB$M_REMOVED>,R7       ;
58   00000452'EF         DE   2B0A            MOVAL   CSB_STATUS_STRNG,R8     ; Point to array of status descriptors.
              FED2       30   2B11            BSBW    DISPLAY_STRING          ; Format the string.
           0180 8F       BA   2B14            POPR    #^M<R7,R8>
                         05   2B18            RSB
                              2B19
```

```
                    2B19          .SBTTL  -       Circuits class display routines.
                    2B19
                    2B19  ;++
                    2B19  ;
                    2B19  ; FUNCTIONAL DESCRIPTION:
                    2B19  ;
                    2B19  ;       This routine is used to display the remote port type using the data
                    2B19  ;       stored at offset PB$L_RPORT_TYP in the path block.  CI780s and CI750s
                    2B19  ;       both have a port type of 2.  If PB$L_RPORT_TYP indicates a port type
                    2B19  ;       code = 2, then the hardware type field in the system block for the
                    2B19  ;       remote system is checked to make the distinction.
                    2B19  ;
                    2B19  ;
                    2B19  ; CALLING SEQUENCE:
                    2B19  ;
                    2B19  ;       JSB     DISPLAY_RPORT_TYP
                    2B19  ;
                    2B19  ; CALLED BY:
                    2B19  ;
                    2B19  ;       GET_DATA
                    2B19  ;
                    2B19  ; INPUTS:
                    2B19  ;
                    2B19  ;       R7  =   SCS offset PB$L_RPORT_TYP from DEFINE FIELD macro.
                    2B19  ;       (AP)[PB_IN]     = Address of path block.
                    2B19  ;       (AP)[SB_IN]     = Address of system block.
                    2B19  ;
                    2B19  ; IMPLICIT INPUTS:
                    2B19  ;
                    2B19  ;       AP      = Address of call frame containing the addresses of the data
                    2B19  ;                 structures to be used when data is extracted.
                    2B19  ;
                    2B19  ; OUTPUTS:
                    2B19  ;
                    2B19  ;       Formated ascii string returned in FORMAT_BUF.
                    2B19  ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                    2B19  ;
                    2B19  ; RETURN STATUS:
                    2B19  ;
                    2B19  ;       R0  =   SS$_NORMAL of failed status returned from FORMAT_DATA.
                    2B19  ;
                    2B19  ;--
                    2B19
                    2B19  DISPLAY_RPORT_TYP:
     0180 8F   BB   2B19          PUSHR   #^M<R7,R8>
     58   04   D0   2B1D          MOVL    #PB_IN,R8               ; Set index into call frame to path
                    2B20                                          ;  block base address.
  57   6C48    C0   2B20          ADDL    (R12)[R8],R7            ; Get the address of the data.
       0216    30   2B24          BSBW    CHECK_DIF               ; Check for change in data since last
                    2B27                                          ;  scan.
         00    EF   2B27          EXTZV   #PB$V_PORT_TYP,-        ; Get the port type code.
 57   67  1F        2B29                  #PB$S_PORT_TYP,(R7),R7  ;
     57   02   C2   2B2C          SUBL    #2,R7                   ; Convert port type code to index into
                    2B2F                                          ;  array of port type descriptors.
          28   12   2B2F          BNEQ    10$                     ; Branch if port is not a CI780 or
                    2B31                                          ;  CI750.
     58   02   D0   2B31          MOVL    #SB_IN,R8               ; Set index into call frame to system
```

```
                            2B34  8302                                              ;  block base address.
          58   6C48    D0   2B34  8303          MOVL    (R12)[R8],R8                ; Get the address of the system block
                            2B38  8304                                              ;  for the remote system.
         58  34  A8    D0   2B38  8305          MOVL    SB$T_HWTYPE(R8),R8          ; Get the remote system hardware type.
   30383756 8F   58    D1   2B3C  8306          CMPL    R8,#^A/V780/                ; Is the remote system an 11/780?
                 14    13   2B43  8307          BEQL    10$                         ; Branch if so.
   35383756 8F   58    D1   2B45  8308          CMPL    R8,#^A/V785/                ; Is the remote system an 11/785?
                 0B    13   2B4C  8309          BEQL    10$                         ; Branch if so.
                 57    D6   2B4E  8310          INCL    R7                          ; Bump port type index.
   30353756 8F   58    D1   2B50  8311          CMPL    R8,#^A/V750/                ; Is the remote system an 11/750?
                 0A    12   2B57  8312          BNEQ    99$                         ; Error if not.
   58    000002CA'EF   DE   2B59  8313 10$:     MOVAL   RPORT_TYP_STRNG,R8          ; Point to array of port type
                            2B60  8314                                              ;  descriptors.
               FE83    30   2B60  8315          BSBW    DISPLAY_STRING              ; Display the port type string.
            0180 8F    BA   2B63  8316 99$:     POPR    #^M<R7,R8>
                 05         2B67  8317          RSB
                            2B68  8318
                            2B68  8319
```

```
                               2B68  8321 ;++
                               2B68  8322 ;
                               2B68  8323 ; FUNCTIONAL DESCRIPTION:
                               2B68  8324 ;
                               2B68  8325 ;       This routine is used to display the state of the virtual circuit.
                               2B68  8326 ;
                               2B68  8327 ; CALLING SEQUENCE:
                               2B68  8328 ;
                               2B68  8329 ;       JSB     DISPLAY_CIR_STATE
                               2B68  8330 ;
                               2B68  8331 ; CALLED BY:
                               2B68  8332 ;
                               2B68  8333 ;       GET_DATA
                               2B68  8334 ;
                               2B68  8335 ; INPUTS:
                               2B68  8336 ;
                               2B68  8337 ;       R7      = SCS Offset PB$W_STATE from DEFINE_FIELD macro.
                               2B68  8338 ;       (AP)[PB_IN]     = Base address of selected path block.
                               2B68  8339 ;
                               2B68  8340 ; IMPLICIT INPUTS:
                               2B68  8341 ;
                               2B68  8342 ;       AP      = Address of call frame containing the addresses of the data
                               2B68  8343 ;                 structures to be used when data is extracted.
                               2B68  8344 ;
                               2B68  8345 ; OUTPUTS:
                               2B68  8346 ;
                               2B68  8347 ;       Formated ascii string returned in FORMAT_BUF.
                               2B68  8348 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                               2B68  8349 ;
                               2B68  8350 ; RETURN STATUS:
                               2B68  8351 ;
                               2B68  8352 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                               2B68  8353 ;
                               2B68  8354 ;--
                               2B68  8355
                               2B68  8356 DISPLAY_CIR_STATE:
            0180 8F     BB     2B68  8357         PUSHR   #^M<R7,R8>
            58   04     D0     2B6C  8358         MOVL    #PB_IN,R8               ; Set index into call frame to path
                               2B6F  8359                                         ;  block base address.
       57   6C48        C0     2B6F  8360         ADDL    (R12)[R8],R7            ; Get address of data.
            01C7        30     2B73  8361         BSBW    CHECK_DIF               ; Check for change in data since last
                               2B76  8362                                         ;  scan.
  58   0000030D'EF      DE     2B76  8363         MOVAL   PB_STATE_STRNG,R8       ; Point to array of circuit state
                               2B7D  8364                                         ;  strings.
       57   67          3C     2B7D  8365         MOVZWL  (R7),R7                 ; Get the state.
  57   8000 8F          B3     2B80  8366         BITW    #PB$C_VC_FAIL,R7        ; Check for VC_FAIL.
            03          13     2B85  8367         BEQLU   10$                     ; Branch if not.
       57   04          B0     2B87  8368         MOVW    #4,R7                   ; Get index to VC_FAIL string.
            FE59        30     2B8A  8369 10$:    BSBW    DISPLAY_STRING          ; Display the string.
            0180 8F     BA     2B8D  8370         POPR    #^M<R7,R8>
            05                 2B91  8371         RSB
                               2B92  8372
                               2B92  8373
```

```
                    2B92  ;++
                    2B92  ;
                    2B92  ; FUNCTIONAL DESCRIPTION:
                    2B92  ;
                    2B92  ;       This routine is used to display the state of the remote port.
                    2B92  ;
                    2B92  ; CALLING SEQUENCE:
                    2B92  ;
                    2B92  ;       JSB     DISPLAY_RSTATE
                    2B92  ;
                    2B92  ; CALLED BY:
                    2B92  ;
                    2B92  ;       GET_DATA
                    2B92  ;
                    2B92  ; INPUTS:
                    2B92  ;
                    2B92  ;       R7      = SCS offset PB$B_RSTATE from DEFINE_FIELD macro.
                    2B92  ;       (AP)[PB_IN]     = Base address of selected path block.
                    2B92  ;
                    2B92  ; IMPLICIT INPUTS:
                    2B92  ;
                    2B92  ;       AP      = Address of call frame containing the addresses of the data
                    2B92  ;                 structures to be used when data is extracted.
                    2B92  ;
                    2B92  ; OUTPUTS:
                    2B92  ;
                    2B92  ;       Formated ascii string returned in FORMAT_BUF.
                    2B92  ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                    2B92  ;
                    2B92  ; RETURN STATUS:
                    2B92  ;
                    2B92  ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                    2B92  ;
                    2B92  ;--
                    2B92
                    2B92  DISPLAY_RSTATE:
      0180 8F   BB  2B92          PUSHR   #^M<R7,R8>
      58   04   D0  2B96          MOVL    #PB_IN,R8               ; Set index into call frame to path
                    2B99                                          ;  block base address.
   57    6C48   C0  2B99          ADDL    (R12)[R8],R7            ; Get the address of the data.
         019D   30  2B9D          BSBW    CHECK_DIF               ; Check for change in data since last
                    2BA0                                          ;  scan.
58  00000330'EF DE  2BA0          MOVAL   PB_RSTATE_STRNG,R8      ; Point to array of state descriptors.
      57   67   9A  2BA7          MOVZBL  (R7),R7                 ; Get the state.
         FE39   30  2BAA          BSBW    DISPLAY_STRING          ; Format the string.
      0180 8F   BA  2BAD          POPR    #^M<R7,R8>
           05       2BB1          RSB
                    2BB2
                    2BB2
```

```
                          2BB2  ;++
                          2BB2  ;
                          2BB2  ; FUNCTIONAL DESCRIPTION:
                          2BB2  ;
                          2BB2  ;       This routine is used to display the path/cable status.  The state of
                          2BB2  ;       the cable is first checked and if good the states of the path A and path
                          2BB2  ;       B are checked.  The output is in the format:
                          2BB2  ;
                          2BB2  ;               "  -  "         Cable is good.  No good paths.
                          2BB2  ;               "A -  "         Cable is good.  Path A is good.
                          2BB2  ;               "  - B"         Cable is good.  Path B is good.
                          2BB2  ;               "A - B"         Cable is good.  Paths A and B are good.
                          2BB2  ;               "CROSSED"       Cable is bad or crossed.
                          2BB2  ;
                          2BB2  ; CALLING SEQUENCE:
                          2BB2  ;
                          2BB2  ;       JSB     DISPLAY_CABLE_STATUS
                          2BB2  ;
                          2BB2  ; CALLED BY:
                          2BB2  ;
                          2BB2  ;       GET_DATA
                          2BB2  ;
                          2BB2  ; INPUTS:
                          2BB2  ;
                          2BB2  ;       R7      = SCS offset PB$B_CBL_STS from DEFINE_FIELD macro.
                          2BB2  ;       (AP)[PB_IN]     = Base address of selected path block.
                          2BB2  ;
                          2BB2  ; IMPLICIT INPUTS:
                          2BB2  ;
                          2BB2  ;       AP      = Address of call frame containing the addresses of the data
                          2BB2  ;                 structures to be used when data is extracted.
                          2BB2  ;
                          2BB2  ; OUTPUTS:
                          2BB2  ;
                          2BB2  ;       Formated ascii string returned in FORMAT_BUF.
                          2BB2  ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                          2BB2  ;
                          2BB2  ; RETURN STATUS:
                          2BB2  ;
                          2BB2  ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                          2BB2  ;
                          2BB2  ;--
                          2BB2
                          2BB2  ASSUME PB$B_P0_STS      EQ PB$B_CBL_STS+1
                          2BB2  ASSUME PB$B_P1_STS      EQ PB$B_P0_STS+1
                          2BB2
                          2BB2  DISPLAY_CABLE_STATUS:
        0380 8F     BB    2BB2          PUSHR   #^M<R7,R8,R9>
          58 04     D0    2BB6          MOVL    #PB_IN,R8               ; Set index into call frame to path
                          2BB9                                          ;  block base address.
     57   6C48      C0    2BB9          ADDL    (R12)[R8],R7            ; Get the address of the data.
          017D      30    2BBD          BSBW    CHECK_DIF               ; Check for change in data since last
                          2BC0                                          ;  scan.
58  0000028C'EF     DE    2BC0          MOVAL   CB_STAT_STRG,R8         ; Point to cable status strings.
       87   01      93    2BC7          BITB    #PB$M_COR_CBL,(R7)+     ; Is the cable status ok?
            05      12    2BCA          BNEQ    10$                     ; Branch if so.
       57   04      D0    2BCC          MOVL    #4,R7                   ; Indicate crossed.
```

```
          1B   11  2BCF  8482           BRB     40$                     ; Skip path state checks.
        0169   30  2BD1  8483 10$:      BSBW    CHECK_DIF               ; Check for change in path A.
          59   D4  2BD4  8484           CLRL    R9                      ; Assume path A and path B bad.
   87     01   93  2BD6  8485           BITB    #PB$M_CUR_PS,(R7)+      ; Path A ok?
          03   13  2BD9  8486           BEQL    20$                     ; Branch if not.
   59     01   C8  2BDB  8487           BISL    #1,R9                   ; Indicate path A ok.
        015C   30  2BDE  8488 20$:      BSBW    CHECK_DIF               ; Check for change in path B.
   67     01   93  2BE1  8489           BITB    #PB$M_CUR_PS,(R7)       ; Path B ok?
          03   13  2BE4  8490           BEQL    30$                     ; Branch if not.
   59     02   C8  2BE6  8491           BISL    #2,R9                   ; Indicate B ok.
   57     59   D0  2BE9  8492 30$:      MOVL    R9,R7                   ; Get result in R7.
        FDF7   30  2BEC  8493 40$:      BSBW    DISPLAY_STRING          ; Display the string.
   0380 8F     BA  2BEF  8494           POPR    #^M<R7,R8,R9>
               05  2BF3  8495           RSB
                   2BF4  8496
                   2BF4  8497
```

```
                          2BF4  ;++
                          2BF4  ;
                          2BF4  ; FUNCTIONAL DESCRIPTION:
                          2BF4  ;
                          2BF4  ;       This routine is used to display the local or remote process names
                          2BF4  ;       associated with a CDT.  The process names, if available, are stored
                          2BF4  ;       following each CDT by the snapshot routine.  The first 16 bytes being
                          2BF4  ;       the local process name and the next 16 bytes being the remote process
                          2BF4  ;       name.  The routine determines which name is to be formatted by the
                          2BF4  ;       offset in R7.
                          2BF4  ;
                          2BF4  ; CALLING SEQUENCE:
                          2BF4  ;
                          2BF4  ;       JSB     DISPLAY_PROC_NAME
                          2BF4  ;
                          2BF4  ; CALLED BY:
                          2BF4  ;
                          2BF4  ;       GET_DATA
                          2BF4  ;
                          2BF4  ; INPUTS:
                          2BF4  ;
                          2BF4  ;       R7      = SCS offset SCS$T_DST_PROC for local process name or
                          2BF4  ;                 SCS$T_SRC_PROC for remote process name.  From DEFINE_FIELD
                          2BF4  ;                 macro.
                          2BF4  ;       (AP)[CDT_IN]    = Base address of selected CDT.
                          2BF4  ;
                          2BF4  ; IMPLICIT INPUTS:
                          2BF4  ;
                          2BF4  ;       AP      = Address of call frame containing the addresses of the data
                          2BF4  ;                 structures to be used when data is extracted.
                          2BF4  ;
                          2BF4  ; OUTPUTS:
                          2BF4  ;
                          2BF4  ;       Formated ascii string returned in FORMAT_BUF.
                          2BF4  ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                          2BF4  ;
                          2BF4  ; RETURN STATUS:
                          2BF4  ;
                          2BF4  ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                          2BF4  ;
                          2BF4  ;--
                          2BF4
                          2BF4  ASSUME SCS$T_SRC_PROC   EQ SCS$T_DST_PROC+16
                          2BF4
                          2BF4  DISPLAY_PROC_NAME:
          0180 8F     BB  2BF4          PUSHR   #^M<R7,R8>
                 50   D4  2BF8          CLRL    R0                      ; Assume failure.
             58  05   D0  2BFA          MOVL    #CDT_IN,R8              ; Set index into call frame to CDT
                          2BFD                                          ;  block base address.
58  0000008C 8F  6C48 C1  2BFD          ADDL3   (R12)[R8],-             ; Get the address of the process names
                          2C06                  #SHC$C_CDT_LENGTH,R8    ;  associated with the CDT by adding
                          2C06                                          ;  the CDT base address to the length
                          2C06                                          ;  of the CDT.
             57  04   D1  2C06          CMPL    #SCS$T_DST_PROC,R7      ; Are we displaying the local process
                          2C09                                          ;  name?
                 08   13  2C09          BEQL    10$                     ; Branch if so.
             57  14   D1  2C0B          CMPL    #SCS$T_SRC_PROC,R7      ; If not, make sure we are displaying
```

```
                        2C0E  8556                                          ;  the remote process name.
            0C    12    2C0E  8557          BNEQ    99$                     ; Error if not.
        58  10    C0    2C10  8558          ADDL    #16,R8                  ; Point to remote process name.
        57  58    D0    2C13  8559 10$:     MOVL    R8,R7                   ; Get address of data in R7.
          0124    30    2C16  8560          BSBW    CHECK_DIF               ; Check for change in data since last
                        2C19  8561                                          ;  scan.
          F963    30    2C19  8562          BSBW    FORMAT_DATA             ; Format the string.
       0180 8F    BA    2C1C  8563 99$:     POPR    #^M<R7,R8>
            05          2C20  8564          RSB
                        2C21  8565
                        2C21  8566
                        2C21  8567 ;++
                        2C21  8568 ;
                        2C21  8569 ; FUNCTIONAL DESCRIPTION:
                        2C21  8570 ;
                        2C21  8571 ;       This routine is used to display the state of a connection.
                        2C21  8572 ;
                        2C21  8573 ; CALLING SEQUENCE:
                        2C21  8574 ;
                        2C21  8575 ;       JSB     DISPLAY_CON_STATE
                        2C21  8576 ;
                        2C21  8577 ; CALLED BY:
                        2C21  8578 ;
                        2C21  8579 ;       GET_DATA
                        2C21  8580 ;
                        2C21  8581 ; INPUTS:
                        2C21  8582 ;
                        2C21  8583 ;       R7      = SCS offset CDT$W_STATE from DEFINE_FIELD macro.
                        2C21  8584 ;       (AP)[CDT_IN]    = Base address of selected CDT.
                        2C21  8585 ;
                        2C21  8586 ; IMPLICIT INPUTS:
                        2C21  8587 ;
                        2C21  8588 ;       AP      = Address of call frame containing the addresses of the data
                        2C21  8589 ;                 structures to be used when data is extracted.
                        2C21  8590 ;
                        2C21  8591 ; OUTPUTS:
                        2C21  8592 ;
                        2C21  8593 ;       Formated ascii string returned in FORMAT_BUF.
                        2C21  8594 ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                        2C21  8595 ;
                        2C21  8596 ; RETURN STATUS:
                        2C21  8597 ;
                        2C21  8598 ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                        2C21  8599 ;
                        2C21  8600 ;--
                        2C21  8601
                        2C21  8602 DISPLAY_CON_STATE:
       0180 8F    BB    2C21  8603          PUSHR   #^M<R7,R8>
        58  05    D0    2C25  8604          MOVL    #CDT_IN,R8              ; Set index into call frame to CDT
                        2C28  8605                                          ;  block base address.
 57   FFFFFFF0 8F  C0   2C28  8606          ADDL    #SHC$L_CDT_DIF,R7       ; Adjust CDT offset to compensate for
                        2C2F  8607                                          ;  portion of CDT not stored.
      57    6C48  C0    2C2F  8608          ADDL    (R12)[R8],R7            ; Get the address of the data.
            0107  30    2C33  8609          BSBW    CHECK_DIF               ; Check for change in data since last
                        2C36  8610                                          ;  scan.
 58   0000035A'EF  DE   2C36  8611          MOVAL   CDT_STATE_STRNG,R8      ; Point to array of connection state
                        2C3D  8612                                          ;  strings.
```

```
57   67  3C 2C3D  8613       MOVZWL  (R7),R7              ; Get the connection state.
   FDA3  30 2C40  8614       BSBW    DISPLAY_STRING       ; Display the string.
0180 8F  BA 2C43  8615       POPR    #^M<R7,R8>
         05 2C47  8616       RSB
            2C48  8617
            2C48  8618
```

```
                              2C48  ;++
                              2C48  ;
                              2C48  ; FUNCTIONAL DESCRIPTION:
                              2C48  ;
                              2C48  ;         This routine is used to display the SCS blocked state.
                              2C48  ;
                              2C48  ; CALLING SEQUENCE:
                              2C48  ;
                              2C48  ;         JSB     DISPLAY_SCS_STATE
                              2C48  ;
                              2C48  ; CALLED BY:
                              2C48  ;
                              2C48  ;         GET_DATA
                              2C48  ;
                              2C48  ; INPUTS:
                              2C48  ;
                              2C48  ;         R7      = SCS offset CDT$W_BLKSTATE from DEFINE_FIELD macro.
                              2C48  ;         (AP)[CDT_IN]    = Base address of selected CDT.
                              2C48  ;
                              2C48  ; IMPLICIT INPUTS:
                              2C48  ;
                              2C48  ;         AP      = Address of call frame containing the addresses of the data
                              2C48  ;                   structures to be used when data is extracted.
                              2C48  ;
                              2C48  ; OUTPUTS:
                              2C48  ;
                              2C48  ;         Formated ascii string returned in FORMAT_BUF.
                              2C48  ;         Length of formatted ascii string returned in FORMAT_BUF_LEN.
                              2C48  ;
                              2C48  ; RETURN STATUS:
                              2C48  ;
                              2C48  ;         R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                              2C48  ;
                              2C48  ;--
                              2C48
                              2C48  DISPLAY_SCS_STATE:
          0180 8F    BB       2C48          PUSHR   #^M<R7,R8>
          58   05    D0       2C4C          MOVL    #CDT_IN,R8              ; Set index into call frame to CDT
                              2C4F                                          ;  block base address.
 57   FFFFFFF0 8F    C0       2C4F          ADDL    #SHC$L_CDT_DIF,R7       ; Adjust CDT offset to compensate for
                              2C56                                          ;  portion of CDT not stored.
          57 6C48    C0       2C56          ADDL    (R12)[R8],R7            ; Get the address of the data.
             00E0    30       2C5A          BSBW    CHECK_DIF               ; Check for change in data since last
                              2C5D                                          ;  scan.
 58   000003C7'EF    DE       2C5D          MOVAL   CDT_BLKSTATE_STRNG,R8   ; Point to array of send blocked state
                              2C64                                          ;  strings.
          57   67    3C       2C64          MOVZWL  (R7),R7                 ; Get the SCS blocked state.
             FD7C    30       2C67          BSBW    DISPLAY_STRING          ; Display the string.
          0180 8F    BA       2C6A          POPR    #^M<R7,R8>
                05            2C6E          RSB
                              2C6F
                              2C6F
                              2C6F          .SBTTL  -       Local port display routines.
```

```
                               2C6F  ;++
                               2C6F  ;
                               2C6F  ; FUNCTIONAL DESCRIPTION:
                               2C6F  ;
                               2C6F  ;       This routine is used to display the kilobyte values CDT$L_KBYTSENT,
                               2C6F  ;       CDT$L_KBYTREQD, and CDT$L_KBYTMAPD.
                               2C6F  ;
                               2C6F  ; CALLING SEQUENCE:
                               2C6F  ;
                               2C6F  ;       JSB     DISPLAY_KBYTES
                               2C6F  ;
                               2C6F  ; CALLED BY:
                               2C6F  ;
                               2C6F  ;       GET_DATA
                               2C6F  ;
                               2C6F  ; INPUTS:
                               2C6F  ;
                               2C6F  ;       R7            = SCS offset from DEFINE_FIELD macro.
                               2C6F  ;       (AP)[CDT_IN]     = Base address of selected CDT.
                               2C6F  ;
                               2C6F  ; IMPLICIT INPUTS:
                               2C6F  ;
                               2C6F  ;       AP            = Address of call frame containing the addresses of the data
                               2C6F  ;                       structures to be used when data is extracted.
                               2C6F  ;
                               2C6F  ; OUTPUTS:
                               2C6F  ;
                               2C6F  ;       Formated ascii string returned in FORMAT_BUF.
                               2C6F  ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                               2C6F  ;
                               2C6F  ; RETURN STATUS:
                               2C6F  ;
                               2C6F  ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                               2C6F  ;
                               2C6F  ;--
                               2C6F
                               2C6F  DISPLAY_KBYTES:
            0180 8F      BB    2C6F          PUSHR   #^M<R7,R8>
             58   05      D0    2C73          MOVL    #CDT_IN,R8              ; Set index into call frame to CDT
                               2C76                                          ;  block base address.
  57   FFFFFFF0 8F      C0    2C76          ADDL    #SHC$L_CDT_DIF,R7       ; Adjust CDT offset to compensate for
                               2C7D                                          ;  portion of CDT not stored.
            57   6C48      C0    2C7D          ADDL    (R12)[R8],R7            ; Get the address of the data.
  67   67   16    0A      EF    2C81          EXTZV   #10,#22,(R7),(R7)       ; Convert bytes to Kbytes.
                 00B4      30    2C86          BSBW    CHECK_DIF               ; Check for change in data since last
                               2C89                                          ;  scan.
                 F8F3      30    2C89          BSBW    FORMAT_DATA             ; Format the string.
            0180 8F      BA    2C8C          POPR    #^M<R7,R8>
                   05      05    2C90          RSB
                               2C91
                               2C91
```

```
                              2C91  ;++
                              2C91  ;
                              2C91  ; FUNCTIONAL DESCRIPTION:
                              2C91  ;
                              2C91  ;       This routine is used to display the status of the port.  The value
                              2C91  ;       at offset MB$L_STS for the selected port in the miscellaneous buffer
                              2C91  ;       is checked for online or offline status.  A string is then placed in
                              2C91  ;       the format buffer which is an ascii representation of the status.
                              2C91  ;
                              2C91  ; CALLING SEQUENCE:
                              2C91  ;
                              2C91  ;       JSB     DISPLAY_PORT_STATUS
                              2C91  ;
                              2C91  ; CALLED BY:
                              2C91  ;
                              2C91  ;       GET_DATA
                              2C91  ;
                              2C91  ; INPUTS:
                              2C91  ;
                              2C91  ;       R7      = Offset MB$L_STS from DEFINE_FIELD macro.
                              2C91  ;       (AP)[MB_IN] = Base address of the data for selected port in MISC_BUF.
                              2C91  ;
                              2C91  ; IMPLICIT INPUTS:
                              2C91  ;
                              2C91  ;       AP      = Address of call frame containing the addresses of the data
                              2C91  ;                 structures to be used when data is extracted.
                              2C91  ;
                              2C91  ; OUTPUTS:
                              2C91  ;
                              2C91  ;       Formated ascii string returned in FORMAT_BUF.
                              2C91  ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                              2C91  ;
                              2C91  ; RETURN STATUS:
                              2C91  ;
                              2C91  ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                              2C91  ;
                              2C91  ;--
                              2C91
                              2C91  DISPLAY_PORT_STATUS:
          0180 8F     BB      2C91          PUSHR   #^M<R7,R8>
            58 07     D0      2C95          MOVL    #MB_IN,R8               ; Set index into call frame to base
                              2C98                                          ;  address of port data in MISC_BUF.
        57   6C48     C0      2C98          ADDL    (R12)[R8],R7            ; Get address of data.
             009F     30      2C9C          BSBW    CHECK_DIF               ; Check for change in data since last
                              2C9F                                          ;  scan.
   58   000002FE'EF   DE      2C9F          MOVAL   PORT_STATUS_STRNG,R8    ; Get address of port status strings.
         03 67   04   E1      2CA6          BBC     #UCB$V_ONLINE,(R7),10$  ; Branch if the port is offline.
         58      88   80      2CAA          ADDB    (R8)+,R8                ; Otherwise, point to online status
                              2CAD                                          ;  string.
         57      58   D0      2CAD  10$:    MOVL    R8,R7                   ; Get address of string.
             F8CC     30      2CB0          BSBW    FORMAT_DATA             ; Format the string.
          0180 8F     BA      2CB3          POPR    #^M<R7,R8>
               05             2CB7          RSB
                              2CB8
                              2CB8
```

```
                            2CB8    ;++
                            2CB8    ;
                            2CB8    ; FUNCTIONAL DESCRIPTION:
                            2CB8    ;
                            2CB8    ;       This routine is used to display the device type of the port.  The value
                            2CB8    ;       at offset MB$L_DEVTYPE is checked for device type of CI780 or CI750.
                            2CB8    ;       The appropriate string is then stored in the format buffer.
                            2CB8    ;
                            2CB8    ; CALLING SEQUENCE:
                            2CB8    ;
                            2CB8    ;       JSB     DISPLAY_PORT_TYPE
                            2CB8    ;
                            2CB8    ; CALLED BY:
                            2CB8    ;
                            2CB8    ;       GET_DATA
                            2CB8    ;
                            2CB8    ; INPUTS:
                            2CB8    ;
                            2CB8    ;       R7      = Offset MB$L_DEVTYPE from DEFINE_FIELD macro.
                            2CB8    ;       (AP)[MB_IN] = Base address of the data for selected port in MISC_BUF.
                            2CB8    ;
                            2CB8    ; IMPLICIT INPUTS:
                            2CB8    ;
                            2CB8    ;       AP      = Address of call frame containing the addresses of the data
                            2CB8    ;                 structures to be used when data is extracted.
                            2CB8    ;
                            2CB8    ; OUTPUTS:
                            2CB8    ;
                            2CB8    ;       Formated ascii string returned in FORMAT_BUF.
                            2CB8    ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                            2CB8    ;
                            2CB8    ; RETURN STATUS:
                            2CB8    ;
                            2CB8    ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                            2CB8    ;
                            2CB8    ;--
                            2CB8
                            2CB8    DISPLAY_PORT_TYPE:
             0180 8F   BB   2CB8            PUSHR   #^M<R7,R8>
               58 07   D0   2CBC            MOVL    #MB_IN,R8               ; Set index into call frame to base
                            2CBF                                            ;  address of port data in MISC_BUF.
          57    6C48   C0   2CBF            ADDL    (R12)[R8],R7            ; Get address of data.
                0077   30   2CC3            BSBW    CHECK_DIF               ; Check for change in data since last
                            2CC6                                            ;  scan.
  58   000002F2'EF     DE   2CC6            MOVAL   DEV_TYP_STRNG,R8        ; Get address of device type strings.
           67     01   D1   2CCD            CMPL    #DT$_CI780,(R7)         ; Is the port a CI780?  ********
                  0C   13   2CD0            BEQL    10$                     ; Branch if so.
           58     88   80   2CD2            ADDB    (R8)+,R8                ; Otherwise point to the next string.
           67     02   D1   2CD5            CMPL    #DT$_CI750,(R7)         ; Is the port a CI750?  ********
                  04   13   2CD8            BEQL    10$                     ; Branch if so.
                  50   D4   2CDA            CLRL    R0                      ; Otherwise indicate failure.
                  06   11   2CDC            BRB     99$                     ; Any return imediately.
           57     58   D0   2CDE    10$:    MOVL    R8,R7                   ; Get address of the string in R7.
                F89B   30   2CE1            BSBW    FORMAT_DATA             ; Format the string.
             0180 8F   BA   2CE4    99$:    POPR    #^M<R7,R8>
                  05        2CE8            RSB
                            2CE9
```

2CE9 8839

```
                          2CE9  ;++
                          2CE9  ;
                          2CE9  ; FUNCTIONAL DESCRIPTION:
                          2CE9  ;
                          2CE9  ;       This routine is used to display the loopback status of the port.
                          2CE9  ;       The value at offsets PDT$B_P0_LBSTS and PDT$B_P1_LBSTS are checked for
                          2CE9  ;       loopback or normal status and converted to an ascii string as follows:
                          2CE9  ;
                          2CE9  ;       "A - B"         -       Loopback tests passed on paths A and B.
                          2CE9  ;       "A -  "         -       Loopback tests passed on path A.
                          2CE9  ;       "  - B"         -       Loopback tests passed on path B.
                          2CE9  ;       "  -  "         -       Loopback tests failed on paths A and B.
                          2CE9  ;       " N/A "         -       Loopback testing is not being done.
                          2CE9  ;
                          2CE9  ;       The ascii string is stored in the format buffer.
                          2CE9  ;
                          2CE9  ; CALLING SEQUENCE:
                          2CE9  ;
                          2CE9  ;       JSB     DISPLAY_LOOPBACK_STATUS
                          2CE9  ;
                          2CE9  ; CALLED BY:
                          2CE9  ;
                          2CE9  ;       GET_DATA
                          2CE9  ;
                          2CE9  ; INPUTS:
                          2CE9  ;
                          2CE9  ;       R7              = Offset PDT$B_P0_LBSTS from DEFINE_FIELD macro.
                          2CE9  ;       (AP)[PDT_IN]    = Base address of the data for selected port in PDT_BUF.
                          2CE9  ;
                          2CE9  ; IMPLICIT INPUTS:
                          2CE9  ;
                          2CE9  ;       AP      = Address of call frame containing the addresses of the data
                          2CE9  ;                 structures to be used when data is extracted.
                          2CE9  ;
                          2CE9  ; OUTPUTS:
                          2CE9  ;
                          2CE9  ;       Formated ascii string returned in FORMAT_BUF.
                          2CE9  ;       Length of formatted ascii string returned in FORMAT_BUF_LEN.
                          2CE9  ;
                          2CE9  ; RETURN STATUS:
                          2CE9  ;
                          2CE9  ;       R0 =    SS$_NORMAL or failed status from FORMAT_DATA.
                          2CE9  ;
                          2CE9  ;--
                          2CE9
                          2CE9  ASSUME  PDT$B_P1_LBSTS  EQ PDT$B_P0_LBSTS+1
             00000002     2CE9  PDT$V_LBDG      =       2
                          2CE9
                          2CE9  DISPLAY_LOOPBACK_STATUS:
            0380 8F   BB  2CE9          PUSHR   #^M<R7,R8,R9>
          58  06      D0  2CED          MOVL    #PDT_IN,R8              ; Set index into call frame to base
                          2CF0                                          ;  address of PDT for this port.
 57   00000110 8F   6C48  C1  2CF0      ADDL3   (R12)[R8],-             ; Get address of local port status.
                          2CF9                  #PDT$W_LPORT_STS,R7     ;
     57   FFFFFF4C 8F   C0  2CF9        ADDL    #SHC$L_PDT_DIF,R7       ; Adjust PDT offset to compensate for
                          2D00                                          ;  portion of PDT not copied.
             003A 30      2D00          BSBW    CHECK_DIF               ; Check for change in data since last
```

```
                         2D03  8898                                          ;  update.
        05 57    02  E0  2D03  8899        BBS     #PDT$V_LBDG,R7,5$         ; Branch if loopback test enabled.
           59    04  D0  2D07  8900        MOVL    #4,R9                     ; Set index to point to ''N/A'' string.
                 1F  11  2D0A  8901        BRB     20$                       ; Display the string.
                 59  D4  2D0C  8902 5$:    CLRL    R9                        ; Assume loopback tests succeeded on
                         2D0E  8903                                          ;  both ports.
 57    00000070  8F  C0  2D0E  8904        ADDL    #<PDT$B_PO_LBSTS--        ; Get address of path A loopback
                         2D15  8905                PDT$W_LPORT_STS>,R7       ;  status.
               0025  30  2D15  8906        BSBW    CHECK_DIF                 ; Check for change in path A loopback
                         2D18  8907                                          ;  status.
           87    02  93  2D18  8908        BITB    #PDT$M_PRV_LBS,(R7)+      ; Path A loopback test succeeded?
                 03  12  2D1B  8909        BNEQ    10$                       ; Branch if so.
           59    02  C8  2D1D  8910        BISL    #2,R9                     ; Index = index + 2.
               001A  30  2D20  8911 10$:   BSBW    CHECK_DIF                 ; Check for change in path B loopback
                         2D23  8912                                          ;  status.
           67    02  93  2D23  8913        BITB    #PDT$M_PRV_LBS,(R7)       ; Path B loopback test suceeded?
                 03  12  2D26  8914        BNEQ    20$                       ; Branch if so.
           59    01  C8  2D28  8915        BISL    #1,R9                     ; Index = index + 1.
           57    59  D0  2D2B  8916 20$:   MOVL    R9,R7                     ; Get index in R7 for DISPLAY_STRING.
 58    000002AC'EF   DE  2D2E  8917        MOVAL   LB_STAT_STRG,R8           ; Point to loopback status strings.
               FCAE  30  2D35  8918        BSBW    DISPLAY_STRING            ; Convert index to string and format.
            0380 8F  BA  2D38  8919        POPR    #^M<R7,R8,R9>
                     05  2D3C  8920        RSB
                         2D3D  8921
                         2D3D  8922
```

```
                         2D3D          .SBTTL  -       CHECK_DIF routine.
                         2D3D   ;++
                         2D3D   ;
                         2D3D   ; FUNCTIONAL DESCRIPTION:
                         2D3D   ;
                         2D3D   ;       This routine uses the address of the data in R7 to determine which
                         2D3D   ;       buffer the data is to be extracted from.  It the uses that address
                         2D3D   ;       to determine where the matching data in the corresponding buffer from
                         2D3D   ;       the previous snapshot can be found.  The data is then compared and
                         2D3D   ;       if found to be is different then a flag is set.  The flag is used by
                         2D3D   ;       the GET_DATA routine to determine when to turn on inverse video.
                         2D3D   ;
                         2D3D   ; CALLING SEQUENCE:
                         2D3D   ;
                         2D3D   ;       JSB     CHECK_DIF
                         2D3D   ;
                         2D3D   ; CALLED BY:
                         2D3D   ;
                         2D3D   ;       DISPLAY_PORT_TYPE
                         2D3D   ;       DISPLAY_PORT_STATUS
                         2D3D   ;       DISPLAY_SCS_STATE
                         2D3D   ;       DISPLAY_CON_STATE
                         2D3D   ;       DISPLAY_PROC_NAME
                         2D3D   ;       DISPLAY_CABLE_STATUS
                         2D3D   ;       DISPLAY_RSTATE
                         2D3D   ;       DISPLAY_CIR_STATE
                         2D3D   ;       DISPLAY_INCN_TIME
                         2D3D   ;       DISPLAY_FIELD
                         2D3D   ;
                         2D3D   ; INPUTS:
                         2D3D   ;
                         2D3D   ;       Address of the data from the most recent snapshot in R7.
                         2D3D   ;
                         2D3D   ; IMPLICIT INPUTS:
                         2D3D   ;
                         2D3D   ;       Data from the previous snapshot stored in OLD_VAR_BUF and OLD_FIX_BUF.
                         2D3D   ;
                         2D3D   ; OUTPUTS:
                         2D3D   ;
                         2D3D   ;       Flag set in MISC_FLAGS if new data is different from the old data.
                         2D3D   ;
                         2D3D   ; RETURN STATUS:
                         2D3D   ;
                         2D3D   ;       None.
                         2D3D   ;
                         2D3D   ; REGISTER USAGE:
                         2D3D   ;
                         2D3D   ;       R2,R3 - Used to calculate the address of the data in the old buffer
                         2D3D   ;       R6 -    Receives the number of bytes of data in the field.  Used to
                         2D3D   ;               to compare the new data with the old.
                         2D3D   ;       R7 -    Contains the address of the data.  Used to calculate the
                         2D3D   ;               address of the data in the old snapshot buffer.
                         2D3D   ;--
                         2D3D
                         2D3D   CHECK_DIF:
        004C 8F    BB    2D3D          PUSHR   #^M<R2,R3,R6>
 000004A3'8F   57  D1    2D41          CMPL    R7,#SNAP_BUF_END        ; Was the data extracted from the fixed
```

```
                              2D48                                          ;  length buffers.
                   71    1F   2D48            BLSSU   40$                   ; Branch if so.
   00000033'EF     57    D1   2D4A            CMPL    R7,CSB_DSCR+4         ; Was the data extracted from the
                              2D51                                          ;  system block buffer?
                   10    1E   2D51            BGEQU   10$                   ; Branch if not.
52     0000002B'EF       D0   2D53            MOVL    SB_DSCR+4,R2          ; Get the address of the system block
                              2D5A                                          ;  buffer.
53     000003FD'EF       D0   2D5A            MOVL    OLD_SB_DSCR+4,R3      ; Get address of system block buffer
                              2D61                                          ;  from previous snapshot.
                   66    11   2D61            BRB     50$                   ; Jump to common code.
   0000003B'EF     57    D1   2D63      10$:  CMPL    R7,PB_DSCR+4          ; Was the data extracted from the
                              2D6A                                          ;  cluster system block buffer?
                   10    1E   2D6A            BGEQU   20$                   ; Branch if not.
52     00000033'EF       D0   2D6C            MOVL    CSB_DSCR+4,R2         ; Get the address of the cluster system
                              2D73                                          ;  block buffer.
53     00000405'EF       D0   2D73            MOVL    OLD_CSB_DSCR+4,R3     ; Get address of cluster system block
                              2D7A                                          ;  buffer from previous snapshot.
                   4D    11   2D7A            BRB     50$                   ; Jump to common code.
   00000043'EF     57    D1   2D7C      20$:  CMPL    R7,CDT_DSCR+4         ; Was the data extracted from the
                              2D83                                          ;  path block buffer?
                   10    1E   2D83            BGEQU   30$                   ; Branch if not.
52     0000003B'EF       D0   2D85            MOVL    PB_DSCR+4,R2          ; Get the address of the path block
                              2D8C                                          ;  buffer.
53     0000040D'EF       D0   2D8C            MOVL    OLD_PB_DSCR+4,R3      ; Get address of path block buffer
                              2D93                                          ;  from previous snapshot.
                   34    11   2D93            BRB     50$                   ; Jump to common code.
56     00000043'EF       D0   2D95      30$:  MOVL    CDT_DSCR+4,R6         ; Get the starting address of the
                              2D9C                                          ;  CDT buffer.
53     0000003F'EF       3C   2D9C            MOVZWL  CDT_DSCR,R3           ; Get the length of the CDT buffer.
              56   53    C0   2DA3            ADDL    R3,R6                 ; Calculate ending address of CDT
                              2DA6                                          ;  buffer.
              56   57    D1   2DA6            CMPL    R7,R6                 ; Was the data extracted from the
                              2DA9                                          ;  CDT buffer?
                   36    1E   2DA9            BGEQU   99$                   ; Branch if not.
52     00000043'EF       D0   2DAB            MOVL    CDT_DSCR+4,R2         ; Get the address of the CDT buffer.
53     00000415'EF       D0   2DB2            MOVL    OLD_CDT_DSCR+4,R3     ; Get address of CDT buffer from
                              2DB9                                          ;  previous snapshot.
                   0E    11   2DB9            BRB     50$                   ; Jump to common code.
52     0000004B'EF       DE   2DBB      40$:  MOVAL   FIX_BUF,R2            ; Get the starting address of the
                              2DC2                                          ;  fixed length buffers.
                              2DC2
53     000004A3'EF       DE   2DC2            MOVAL   OLD_FIX_BUF,R3        ; Get the starting address of the fixed
                              2DC9                                          ;  length buffers from the previous
                              2DC9                                          ;  snapshot.
         52   57   52    C3   2DC9      50$:  SUBL3   R2,R7,R2              ; Get the offset into the buffer where
                              2DCD                                          ;  the data is located.
              53   52    C0   2DCD            ADDL    R2,R3                 ; Get the corresponding address of the
                              2DD0                                          ;  data from the previous snapshot.
           56   04 AA    9B   2DD0            MOVZBW  FD$B_LENGTH(R10),R6   ; Get the number of bytes of raw data.
         63   67   56    29   2DD4            CMPC    R6,(R7),(R3)          ; Compare the old data with the new.
                   07    13   2DD8            BEQL    99$                   ; Branch if they're the same.
   00000476'EF     04    C8   2DDA            BISL    #MF$M_INVERSE,MISC_FLAGS; Set the inverse video flag.
           004C 8F       BA   2DE1      99$:  POPR    #^M<R2,R3,R6>
                         05   2DE5            RSB
                              2DE6
                              2DE6
```

```
2DE6          .SBTTL  Description of snapshot buffers.
2DE6    ;
2DE6    ;SNAP_BUF:       Start of buffers to be locked.
2DE6    ;
2DE6    ;                                 +--------+
2DE6    ;CURRENT_REPORT:                  |        |   Bit 0 set indicates cluster report.
2DE6    ;                                 +--------+
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;CLASS_SELECT:                    | USER selected classes for display |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+--------+--------+
2DE6    ;LOCSYSID:                        |          ID number of the local system.             |
2DE6    ;                                 +--------+--------+--------+--------+--------+--------+
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;NUM_BUF_DSCR:                    |  # of buffer descriptors in use   |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;NUM_FREE_BUF_DSCR:               | # of buffer descriptors available |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;LOC_SB_ADRS:                     |   Address of local system block   |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;SB_BUF_PNTR:                     |       Pointer into SB buffer      |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;CSB_BUF_PNTR:                    |      Pointer into CSB buffer      |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;PB_BUF_PNTR:                     |      Pointer into PB buffer       |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;CDT_BUF_PNTR:                    |      Pointer into CDT buffer      |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ; VAR_BUF_ADRS:                   |Address of variable length buffers |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;
2DE6    ; Descriptors for the variable length snapshot buffers.
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;SB_DSCR:                         |#entrys in buffer|Length of buffer |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;                                 |         Address of buffer         |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;CSB_DSCR:                        |#entrys in buffer|Length of buffer |
2DE6    ;                                 +--------+--------+--------+--------+
2DE6    ;                                 |         Address of buffer         |
```

```
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;PB_DSCR:                             |#entrys in buffer|Length of buffer |
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;                                     |        Address of buffer          |
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;CDT_DSCR:                            |#entrys in buffer|Length of buffer |
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;                                     |        Address of buffer          |
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;
2DE6  ; Buffer for system blocks.
2DE6  ;
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;          ^ SB$L_FLINK               |  Pointer to next system block     |
2DE6  ;          |                          +--------+--------+--------+--------+
2DE6  ;          | SB$L_BLINK:              |         SB$SHC$L_NUM_PB           |
2DE6  ;          |                          +--------+--------+--------+--------+
2DE6  ;          |                          |                 |SHC$SB$B|        |
2DE6  ;          |                          |                 |_DIS_POS|        |
2DE6  ;          |                          +--------+--------+--------+--------+
2DE6  ;     SB$C_LENGTH                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;          |                          |                                   |
2DE6  ;          |                          |                                   |
2DE6  ;          |                          |                                   |
2DE6  ;          |                          +--------+--------+--------+--------+
2DE6  ;          V   SB$L_CSB:              | Pointer to CSB for this system    |
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;                                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;                                     ~r~        More system blocks       ~
2DE6  ;                                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;
2DE6  ;
2DE6  ; Buffer for cluster system blocks.
2DE6  ;
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;                 ^                   |                                   |
2DE6  ;                 |                   |                                   |
2DE6  ;                 |                   |                                   |
2DE6  ;           CSB$C_LENGTH              |     One cluster system block      |
2DE6  ;                 |                   |                                   |
2DE6  ;                 |                   |                                   |
2DE6  ;                 V                   |                                   |
2DE6  ;                                     +--------+--------+--------+--------+
2DE6  ;                                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;                                     |                                   |
2DE6  ;                                     ~                                   ~
2DE6  ;                                         More cluster system blocks
```

```
2DE6   ;                              |                                       |
2DE6   ;                              |                                       |
2DE6   ;                              |                                       |
2DE6   ;                              +---------+---------+---------+---------+
2DE6   ;
2DE6   ;
2DE6   ; Buffer for path blocks.
2DE6   ;
2DE6   ;                              +---------+---------+---------+---------+
2DE6   ;                    ^         |                 Flink                 |
2DE6   ;                    |         +---------+---------+---------+---------+
2DE6   ;                    |         |           PB$SHC$L_NUM_CDT            |
2DE6   ;                    |         +---------+---------+---------+---------+
2DE6   ;          PB$C_LENGTH         |                                       |
2DE6   ;                              |                                       |
2DE6   ;                    |         |            One path block             |
2DE6   ;                    |         |                                       |
2DE6   ;                    |         |                                       |
2DE6   ;                    V         |                                       |
2DE6   ;                              +---------+---------+---------+---------+
2DE6   ;                              |                                       |
2DE6   ;                              |                                       |
2DE6   ;                              |                                       |
2DE6   ;                              -                                       -
2DE6   ;                              |           More path blocks            |
2DE6   ;                              |                                       |
2DE6   ;                              |                                       |
2DE6   ;                              +---------+---------+---------+---------+
2DE6   ;
2DE6   ; Buffer for CDT's.
2DE6   ;
2DE6   ;                              +---------+---------+---------+---------+
2DE6   ;                    ^         |                FLINK                  |
2DE6   ;                    |         +---------+---------+---------+---------+
2DE6   ;                    |         |                                       |
2DE6   ;                              |                                       |
2DE6   ;    SHC$C_CDT_LENGTH+32       |               One CDT                 |
2DE6   ;                              |     (portion of actual CDT from       |
2DE6   ;                    |         |     CDT$L_RCONID to CDT$W_DGREC)      |
2DE6   ;                    |         |                                       |
2DE6   ;                    V         |                                       |
2DE6   ;                              +---------+---------+---------+---------+
2DE6   ;                              |                                       |
2DE6   ;                              |                                       |
2DE6   ;                              -                                       -
2DE6   ;                              |           Lots more CDT's             |
2DE6   ;                              |                                       |
2DE6   ;                              |                                       |
2DE6   ;                              |                                       |
2DE6   ;                              +---------+---------+---------+---------+
2DE6   ;
2DE6   ; Fixed length buffers
2DE6   ;                              +---------+---------+---------+---------+
2DE6   ;CLUB_BUF:           ^         |                                       |
2DE6   ;                    |         |                                       |
2DE6   ;         CLUB$C_LENGTH        |         Buffer for the CLUB           |
2DE6   ;                    |         |                                       |
```

```
2DE6  ;                  !         !                                       !
2DE6  ;                  V         +---------+---------+---------+---------+
2DE6  ;                            +---------+---------+---------+---------+
2DE6  ;DUMMY_SB:         ^         !                                       !
2DE6  ;                  !         !                                       !
2DE6  ;                  !         !                                       !
2DE6  ;                            !                                       !
2DE6  ;              SB$C_LENGTH   !        One fake system block          !
2DE6  ;                            !                                       !
2DE6  ;                  !         !                                       !
2DE6  ;                  !         !                                       !
2DE6  ;                  V         !                                       !
2DE6  ;                            +---------+---------+---------+---------+
2DE6  ;DUMMY_PB:         ^         !                                       !
2DE6  ;                  !         !                                       !
2DE6  ;                  !         !                                       !
2DE6  ;                            !                                       !
2DE6  ;              PB$C_LENGTH   !         One fake path block           !
2DE6  ;                            !                                       !
2DE6  ;                  !         !                                       !
2DE6  ;                  !         !                                       !
2DE6  ;                  V         !                                       !
2DE6  ;                            +---------+---------+---------+---------+
2DE6  ;
2DE6  ; Buffer for PDT's.
2DE6  ;
2DE6  ;                            +---------+---------+---------+---------+
2DE6  ;PDT_BUF: ^          ^       !                                       !
2DE6  ;         !          !       !                                       !
2DE6  ;         !          !       !                                       !
2DE6  ;                    !       !              First PDT                !
2DE6  ; SHC$L_PDT_LENGTH   !       !    (portion of actual PDT from        !
2DE6  ;                    !       !   PDT$L_MSGHDRSZ to PDT$L_LBDG)       !
2DE6  ;         !          !       !                                       !
2DE6  ;         !          !       !                                       !
2DE6  ;         V          !       !                                       !
2DE6  ;                    !       +---------+---------+---------+---------+
2DE6  ;                    !       !                                       !
2DE6  ;                    !       !                                       !
2DE6  ;SHC$L_PDT_LENGTH*MAXPDT     !                                       !
2DE6  ;                            !                                       !
2DE6  ;                    !       -             Second PDT                -
2DE6  ;                    !                                                
2DE6  ;                    !       !                                       !
2DE6  ;                    !       !                                       !
2DE6  ;                    !       !                                       !
2DE6  ;                    V       !                                       !
2DE6  ;                            +---------+---------+---------+---------+
2DE6  ;
2DE6  ;
2DE6  ; Buffer for miscellaneous port data.
2DE6  ;
2DE6  ;
2DE6  ;                            +---------+---------+---------+---------+
2DE6  ;MISC_BUF:                   !            MB$L_PORT_NAME             !
2DE6  ;                            +---------+---------+---------+---------+
2DE6  ;                            !            MB$L_PORT_NUM              !
2DE6  ;                            +---------+---------+---------+---------+
```

```
2DE6 ;                                     |            MB$L_DEVTYPE           |
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;                                     |              MB$L_STS             |
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;                                     |   MB$W_ERRCNT   | ERTMAX | ERTCNT |
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;                                     |            MB$L_FORM_PB           |
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;                                     |          MB$L_PL_WAITERS          |
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;                                     |  MB$W_MSG_FREE  |   MB$W_DG_FREE  |
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;                                     | MB$W_RSPS_QUED  | MB$W_CMDS_QUED  |
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;MISC_BUF+MISC_BUF_LEN:               |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     |     MISC_BUF for second port      |
2DE6 ;                                     |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     |                                   |
2DE6 ;                                     +--------+--------+--------+--------+
2DE6 ;
2DE6 ;
2DE6 ;
2DE6 ;SNAP_BUF_END:                        End of buffers to be locked.
2DE6 ;
2DE6 ;
2DE6
2DE6
```

```
                        2DE6  9302            .SBTTL  Snapshot buffers.
                    00000000  9303            .PSECT  SNAP_DATA,RD,WRT,NOEXE
                        0000  9304
                        0000  9305  SNAP_BUF:                                  ; Start of buffers to be locked.
                        0000  9306
                        0000  9307  CURRENT_REPORT:                            ; Indicates which report the user has
                  00    0000  9308            .BYTE   0                        ;  selected for display.
                        0001  9309
                        0001  9310  CLASS_SELECT:                              ; Indicates which classes the user has
            00000000    0001  9311            .LONG   0                        ;  selected for display.
                        0005  9312
                        0005  9313  LOCSYSID:                                  ; ID number of the local system.
            0000000B    0005  9314            .BLKB   6
                        000B  9315
                        000B  9316  NUM_BUF_DSCR:                              ; Number of buffer descriptors in use.
            00000000    000B  9317            .LONG   0
                        000F  9318
                        000F  9319  NUM_FREE_BUF_DSCR:                         ; Number of buffer descriptors
            00000000    000F  9320            .LONG   0                        ;  available.
                        0013  9321
                        0013  9322  LOC_SB_ADRS:                               ; Address of system block for local
            00000000    0013  9323            .LONG   0                        ;  system.
                        0017  9324
                        0017  9325  SB_BUF_PNTR:                               ; Pointer into system block buffer.
            00000000    0017  9326            .LONG   0                        ;
                        001B  9327  CSB_BUF_PNTR:                              ; Pointer into cluster system block
            00000000    001B  9328            .LONG   0                        ;  buffer.
                        001F  9329  PB_BUF_PNTR:                               ; Pointer in path block buffer.
            00000000    001F  9330            .LONG   0                        ;
                        0023  9331  CDT_BUF_PNTR:                              ; Pointer into CDT buffer.
            00000000    0023  9332            .LONG   0                        ;
                        0027  9333
   00000000 00000000    0027  9334  SB_DSCR:.QUAD   0                          ; System block buffer.
                        002F  9335
                        002F  9336  CSB_DSCR:                                  ; Cluster system block buffer.
   00000000 00000000    002F  9337            .QUAD   0                        ;
                        0037  9338
   00000000 00000000    0037  9339  PB_DSCR:.QUAD   0                          ; Path block buffer.
                        003F  9340
                        003F  9341  CDT_DSCR:                                  ; Connection descriptor table buffer.
   00000000 00000000    003F  9342            .QUAD   0                        ;
                        0047  9343
                        0047  9344  VAR_BUF_ADRS:                              ; Copy of the variable buffer address
            00000000    0047  9345            .LONG   0                        ;  returned from GET_VM.
                        004B  9346
                        004B  9347  FIX_BUF:                                   ; Start of fixed length buffers.
                        004B  9348
                        004B  9349  CLUB_BUF:                                  ; Buffer for the club.
            000001F3    004B  9350            .BLKB   CLUB$C_LENGTH            ;
                        01F3  9351
                        01F3  9352  DUMMY_SB:                                  ; Fake system block for formative path
            00000253    01F3  9353            .BLKB   SB$C_LENGTH              ;  blocks without an associated
                        0253  9354                                             ;  formative system block.
                        0253  9355
                        0253  9356  DUMMY_PB:                                  ; Fake path block for connections in
            000002A7    0253  9357            .BLKB   PB$C_LENGTH              ;  the listen state.
                        02A7  9358
```

```
          02A7  PDT_BUF:
00000447  02A7          .BLKB   SHC$L_PDT_LENGTH*MAXPDT ; Buffer for port information.
          0447
          0447  MISC_BUF:                               ; Buffer for miscellaneous information.
000004A3  0447          .BLKB   MISC_BUF_LEN*2
          04A3
          04A3  SNAP_BUF_END:                           ; End of buffers to be locked.
          04A3
00000458  04A3  FIX_BUF_LEN     = .-FIX_BUF
000004A3  04A3  SNAP_BUF_LEN    = .-SNAP_BUF
          04A3
          04A3
```

```
    04A3          .SBTTL  Snapshot routines.
00002DE6          .PSECT  MAIN,RD,EXE,NOWRT
    2DE6    ;++
    2DE6    ;
    2DE6    ; FUNCTIONAL DESCRIPTION:
    2DE6    ;
    2DE6    ;       This routine is used to extract the data from the SCS database and the
    2DE6    ;       CI port database.  This routine is called whenever the interval timer
    2DE6    ;       expires or upon completion of parsing and executing a command.
    2DE6    ;
    2DE6    ;
    2DE6    ; CALLING SEQUENCE:
    2DE6    ;
    2DE6    ;       $CMKRNL_S -
    2DE6    ;               ROUTINE=SNAPSHOT
    2DE6    ;
    2DE6    ; CALLED BY:
    2DE6    ;
    2DE6    ;       DO_SNAPSHOT
    2DE6    ;
    2DE6    ;
    2DE6    ; IMPLICIT IMPUTS:
    2DE6    ;
    2DE6    ;       Cluster and port data stored in SCS database and CI port database.
    2DE6    ;
    2DE6    ;
    2DE6    ; IMPLICIT OUTPUTS:
    2DE6    ;
    2DE6    ;       Extracted data stored in snapshot buffers.  For details see previous
    2DE6    ;       section.
    2DE6    ;
    2DE6    ;
    2DE6    ; RETURN STATUS:
    2DE6    ;
    2DE6    ;       SS$_NORMAL or SS$_NOSUCHNODE returned in R0.
    2DE6    ;
    2DE6    ;
    2DE6    ; SIDE EFFECTS:
    2DE6    ;
    2DE6    ;       It should be noted that this routine runs at IPL$_SCS and calls a
    2DE6    ;       routine CHECK_QUE that runs at IPL 31.  Also both routines and all
    2DE6    ;       the buffers used by the routines are locked in memory during execution
    2DE6    ;       of this routine.
    2DE6    ;
    2DE6    ;
    2DE6    ; REGISTER USAGE:
    2DE6    ;
    2DE6    ;       Listed here is a general register usage.  Detailed usage is given in
    2DE6    ;       each subsection.
    2DE6    ;
    2DE6    ;       R3  -   Used as destination pointer when storing structures in buffers.
    2DE6    ;       R4  -   Used as a temporary register and by MOVC's.
    2DE6    ;       R5  -   Used as a temporary register and by MOVC's.
    2DE6    ;       R6  -   Used as a temporary register.
    2DE6    ;       R7  -   Used to store the address of the first link in a doubly linked
    2DE6    ;               list.
    2DE6    ;       R8  -   Used as a temporary register and pointer to the base address to
```

```
                              2DE6  ;                 store data in MISC_BUF.
                              2DE6  ;         R9  -   Used as a temporary register and to save a copy of R3 before
                              2DE6  ;                 R3 is updated to point to next slot in buffer.
                              2DE6  ;         R10 -   Used to store the address of the next link in a doubly linked
                              2DE6  ;                 list.
                              2DE6  ;             -   Used as source pointer when storing structures in buffers.
                              2DE6  ;--
                              2DE6
                        07F8  2DE6  .ENTRY  SNAPSHOT,^M<R3,R4,R5,R6,R7,R8,R9,R10>
                    00    E1  2DE8          BBC     #CLS$V_CLUB,-           ; Branch if display of the club is
           16 00000001'EF     2DEA                  CLASS_SELECT,5$         ;  disabled.
   5A      00000000'GF    D0  2DF0          MOVL    G^CLU$GL_CLUB,R10       ; Get address of the club.
                    0D    13  2DF7          BEQL    5$                      ; Branch if there isn't a club.
   53      0000004B'EF    DE  2DF9          MOVAL   CLUB_BUF,R3             ; Get address to store the club.
          6A    01A8 8F   28  2E00          MOVC3   #CLUB$C_LENGTH,(R10),-  ; Store club in club buffer.
                    63        2E05                  (R3)                    ;
   50      0000028C 8F    D0  2E06  5$:     MOVL    #SS$_NOSUCHNODE,R0      ; Assume that there is no cluster.
           00000000'GF    D1  2E0D          CMPL    G^SCS$GQ_CONFIG,-       ; Any remote SB's known?
           00000000'8F        2E13                  #SCS$GQ_CONFIG          ;
                              2E18          JEQL    SNAPSHOT_ERR            ; If not, exit with error status.
                              2E1D  10$:    DSBINT  #IPL$_SCS               ; Lock the SCS database.
                    00    E0  2E23          BBS     #RPT$V_CLSTR,-          ; Branch to cluster snapshot routine
           00000000'EF        2E25                  CURRENT_REPORT,-        ;  if we're displaying the cluster
                    03        2E2A                  CLUSTER_SNAP            ;  report.
                  02B1    31  2E2B          BRW     LOCAL_SNAP              ; Otherwise branch to the local
                              2E2E                                          ;  snapshot routine.
                              2E2E
                              2E2E  ;
                              2E2E  ; Store the information necessary to display the cluster report.  This includes
                              2E2E  ; all system blocks, path blocks, and connection descriptors.  Formative path
                              2E2E  ; blocks and connections descriptors for connections in the listen state will
                              2E2E  ; also be stored if the user has elected to see them.
                              2E2E  ;
                              2E2E
                              2E2E  CLUSTER_SNAP:
           00000013'EF    D4  2E2E          CLRL    LOC_SB_ADRS             ; Clear address of local system block.
           000001FF'EF    D4  2E34          CLRL    DUMMY_SB+SB$L_PBFL      ; Assume no formative path blocks with
                              2E3A                                          ;  formative system blocks.
           00000253'EF    D4  2E3A          CLRL    DUMMY_PB                ; Assume there are no additional
                              2E40                                          ;  formative path blocks with
                              2E40                                          ;  associated system blocks.
                              2E40
                              2E40  ;
                              2E40  ; Store each system block in a singly linked list in SB buffer.
                              2E40  ;
                              2E40  ; REGISTER USAGE:
                              2E40  ;
                              2E40  ;         R3  -   Pointer into SB buffer where the system block will be stored.
                              2E40  ;         R9  -   Used to save a copy of R3 to be used after R3 is updated to
                              2E40  ;                 point to the next free slot in SB buffer.
                              2E40  ;         R10 -   Pointer to system block in the SCS database.
                              2E40  ;
                              2E40
   53      00000017'EF    D0  2E40  SB_SNAP:MOVL    SB_BUF_PNTR,R3          ; Get address of system block buffer.
   5A      00000000'GF    D0  2E47          MOVL    G^SCS$GQ_CONFIG,R10     ; Get address of first system block.
            18 AA    2F   E1  2E4E  10$:    BBC     #47,SB$B_SYSTEMID(R10),-; Branch if not illegal type or system
                    34        2E52                  SB_FIRST_LOOP           ;  block for UNA.
```

```
             5A   6A   D0 2E53           MOVL    (R10),R10               ; Get address of next system block.
   00000000'8F   5A   D1 2E56           CMPL    R10,#SCS$GQ_CONFIG      ; Any more system blocks?
                           2E5D          JEQL    SB_LOOP_END             ; Branch if not.
                  EA   11 2E62           BRB     10$                     ; Otherwise, loop for all system blocks.
                           2E64  ;
                           2E64  ; The following is skipped on the first pass because the first system block has
                           2E64  ; no previous system block to link to.
                           2E64  ;
                           2E64
       00000029'EF   B7 2E64  SB_LOOP:DECW   SB_DSCR+2              ; Anymore room left in the buffer?
                           2E6A          JEQLU   SB_LOOP_END             ; Branch if not.
             5A   6A   D0 2E6F  10$:    MOVL    (R10),R10               ; Get address of next system block.
   00000000'8F   5A   D1 2E72           CMPL    R10,#SCS$GQ_CONFIG      ; Any more system blocks?
                           2E79          JEQL    SB_LOOP_END             ; Branch if not.
        18 AA   2F   E0 2E7E           BBS     #47,SB$B_SYSTEMID(R10),-; Ignore illegal and UNA system blocks.
                  EC      2E82                   10$                     ;
        A0 A3   53   D0 2E83           MOVL    R3,-SB$C_LENGTH(R3)     ; Link previous system block to this
                           2E87                                          ;  one.
                           2E87  SB_FIRST_LOOP:
             59   53   D0 2E87           MOVL    R3,R9                   ; Save address where this SB will be
                           2E8A                                          ;  stored.
 63    6A   0060 8F   28 2E8A           MOVC3   #SB$C_LENGTH,(R10),(R3) ; Store system block in SB buffer.
        A0 A3        D4 2E90           CLRL    -SB$C_LENGTH(R3)        ; Assume no more system blocks.
                           2E93  ;
                           2E93  ; Save the address of the local system block.  This will be used by the
                           2E93  ; LISTENER_SNAP routine to link a dummy path block to the local system block
                           2E93  ; if any listeners are found.
                           2E93  ;
                           2E93
       00000013'EF   D5 2E93           TSTL    LOC_SB_ADRS             ; Have we already saved the address of
                           2E99                                          ;  the local system block?
                  1B   12 2E99           BNEQ    CSB_SNAP                ; Branch if we have.
               18 A9   D1 2E9B           CMPL    SB$B_SYSTEMID(R9),-     ; Is this system block for the local
       00000005'EF        2E9E                   LOC$YSID                ; system?
                  11   12 2EA3           BNEQ    CSB_SNAP                ; Branch if not.
               1C A9   B1 2EA5           CMPW    SB$B_SYSTEMID+4(R9),-   ; Must check all six bytes of ID.
       00000009'EF        2EA8                   LOC$YSID+4              ;
                  07   12 2EAD           BNEQ    CSB_SNAP                ; Branch if no match.
   00000013'EF   59   D0 2EAF           MOVL    R9,LOC_SB_ADRS          ; Save address of local system block.
                           2EB6
                           2EB6
                           2EB6  ;
                           2EB6  ; This code executes if the display of Cluster System Blocks is enabled.
                           2EB6  ; If there is a CSB associated with the last System Block stored in the SB
                           2EB6  ; buffer, then it stores the CSB in the CBS buffer and adjusts SB$L_CSB to
                           2EB6  ; point to the CSB.
                           2EB6  ;
                           2EB6  ; INPUTS:
                           2EB6  ;         R3,R9   - Point to SB buffer where last SB was stored.
                           2EB6  ;
                           2EB6  ; REGISTER USAGE:
                           2EB6  ;
                           2EB6  ;         R3      - Used to point to CSB buffer where a CSB will be stored.
                           2EB6  ;         R9      - Pointer to last SB stored the SB_BUFFER.
                           2EB6  ;
                           2EB6
                           2EB6  CSB_SNAP:
```

```
                02   E1  2EB6  9543            BBC     #CLS$V_CSB,-                 ; Skip saving of CSB if MEMBERS class
   26 00000001'EF        2EB8  9544                    CLASS_SELECT,PB_SNAP         ;  is disabled.
           0408 8F   BB  2EBE  9545            PUSHR   #^M<R3,R10>
        5A    5C A9  D0  2EC2  9546            MOVL    SB$L_CSB(R9),R10             ; Get the address of the Cluster System
                         2EC6  9547                                                 ;  Block for this system.
                18   13  2EC6  9548            BEQL    CSB_SNAP_END                 ; Branch if no CSB.
   53    0000001B'EF D0  2EC8  9549            MOVL    CSB_BUF_PNTR,R3              ; Get the address to store the CSB in
                         2ECF  9550                                                 ;  the CSB buffer.
        5C A9    53  D0  2ECF  9551            MOVL    R3,SB$L_CSB(R9)              ; Save pointer to CSB in SB.
      6A   00AC 8F   28  2ED3  9552            MOVC3   #CSB$C_LENGTH,(R10),-        ; Store the CSB in the CSB buffer.
                63       2ED8  9553                    (R3)                         ;
  0000001B'EF    53  D0  2ED9  9554            MOVL    R3,CSB_BUF_PNTR              ; Save address to store next CSB.
                         2EE0  9555
                         2EE0  9556 CSB_SNAP_END:
           0408 8F   BA  2EE0  9557            POPR    #^M<R3,R10>
                         2EE4  9558
                         2EE4  9559 ;
                         2EE4  9560 ; Store the path blocks in singly linked lists in PB buffer. Link each list to
                         2EE4  9561 ; the associated system block by saving the address of the first path block
                         2EE4  9562 ; at offset SB$L_PBFL in the system block.
                         2EE4  9563 ;
                         2EE4  9564 ; Note: This loop does not store formative path blocks. See FORM_PB_SNAP:.
                         2EE4  9565 ;
                         2EE4  9566 ; INPUTS:
                         2EE4  9567 ;
                         2EE4  9568 ;          R9  -   Address of system block SB buffer that is associated with this
                         2EE4  9569 ;                  path block.  (Passed from SB_SNAP routine)
                         2EE4  9570 ;          R10 -   Address of system block in SCS database for the last system
                         2EE4  9571 ;                  block stored in SB buffer.
                         2EE4  9572 ;
                         2EE4  9573 ; REGISTER USAGE:
                         2EE4  9574 ;
                         2EE4  9575 ;          R3  -   Pointer into PB buffer where the path block will be stored.
                         2EE4  9576 ;          R7  -   Address of link to first path block on the system block.
                         2EE4  9577 ;          R9  -   Used to save a copy of R3 to be used after R3 is updated to
                         2EE4  9578 ;                  point to the next free slot in PB buffer.
                         2EE4  9579 ;          R10 -   Pointer to path block in the SCS database.
                         2EE4  9580 ;
                         2EE4  9581
           0688 8F   BB  2EE4  9582 PB_SNAP:PUSHR  #^M<R3,R7,R9,R10>
             0C A9   D4  2EE8  9583            CLRL    SB$L_PBFL(R9)                ; Assume no paths on this system block.
   53    0000001F'EF D0  2EEB  9584            MOVL    PB_BUF_PNTR,R3               ; Get address of first free slot in
                         2EF2  9585                                                 ; path block buffer.
      57    0C AA    DE  2EF2  9586            MOVAL   SB$L_PBFL(R10),R7            ; Get address of forward link to first
                         2EF6  9587                                                 ;  path block?
      57    0C AA    D1  2EF6  9588            CMPL    SB$L_PBFL(R10),R7            ; Are there any paths on this system
                         2EFA  9589                                                 ;  block?
                         2EFA  9590            JEQL    PB_LOOP_END                  ; If not, exit this loop.
      0C A9      53  D0  2EFF  9591            MOVL    R3,SB$L_PBFL(R9)             ; Save address of first path block in
                         2F03  9592                                                 ;  associated system block.
         5A      57  D0  2F03  9593            MOVL    R7,R10                       ; Get a copy of link address in R10.
         5A      6A  D0  2F06  9594            MOVL    (R10),R10                    ; Get forward link to the first path
                         2F09  9595                                                 ;  block.
                1A   11  2F09  9596            BRB     PB_FIRST_LOOP                ; Go save the first path block in
                         2F0B  9597                                                 ;  PB buffer.
                         2F0B  9598
                         2F0B  9599 PB_LOOP:
```

```
   00000039'EF   B7 2F0B        DECW    PB_DSCR+2                  ; Any room left in the path block
                    2F11                                           ;  buffer?
                    2F11        JEQLU   PB_LOOP_END                ; Branch if not.
          5A  6A D0 2F16        MOVL    (R10),R10                  ; Get forward link to the (next) path
                    2F19                                           ;  block.
          57  5A D1 2F19        CMPL    R10,R7                     ; Does the forward link point back to
                    2F1C                                           ;  first forward link?
                    2F1C        JEQL    PB_LOOP_END                ; If so, no more path blocks.
       AC A3  53 D0 2F21        MOVL    R3,-PB$C_LENGTH(R3)        ; Link previous path block to this one.
                    2F25
                    2F25 PB_FIRST_LOOP:
          59  53 D0 2F25        MOVL    R3,R9                      ; Save address where this path block
                    2F28                                           ;  will be stored.
 63  6A  0054 8F 28 2F28        MOVC3   #PB$C_LENGTH,(R10),(R3)    ; Copy path block to PB buffer.
          AC A3  D4 2F2E        CLRL    -PB$C_LENGTH(R3)           ; Assume no more path blocks.
          E0 A3  D4 2F31        CLRL    -PB$C_LENGTH+PB$L_CDTLST(R3)
                    2F34                                           ; Assume no CDT's on this path block.
          B0 A3  D4 2F34        CLRL    -PB$C_LENGTH+PB$SHC$L_NUM_CDT(R3)
                    2F37 ;
                    2F37 ; Get the number of SCS waiters on each path block by counting the entries
                    2F37 ; in the queue at offset PB$L_WAITQFL.  Save the count at the same offset in
                    2F37 ; place of the forward link (since it won't be used again)
                    2F37 ;
                    2F37 ; Fix the port name to be PA%0 rather than PA%"null".
                    2F37 ;
                    2F37 ; INPUTS:
                    2F37 ;
                    2F37 ;          R9  -   Address of path block that was last stored in PB buffer.
                    2F37 ;                  (Passed from PB_SNAP routine)
                    2F37 ;          R10 -   Address of path block in the SCS database for the last path
                    2F37 ;                  block stored in PB buffer.
                    2F37 ;
                    2F37 ; REGISTER USAGE:
                    2F37 ;
                    2F37 ;          R7  -   Address of link to first entry in SCS wait queue.
                    2F37 ;          R10 -   Pointer to entry in SCS wait queue.
                    2F37 ;
                    2F37
                    2F37 SCS_WAITER_SNAP:
        0480 8F  BB 2F37        PUSHR   #^M<R7,R10>
    57   38 AA   DE 2F3B        MOVAL   PB$L_WAITQFL(R10),R7       ; Get the address of the forward link
                    2F3F                                           ;  for the first entry in the SCS
                    2F3F                                           ;  wait queue.
          5A  57 D0 2F3F        MOVL    R7,R10                     ; Get a copy in R10.
                    2F42
                    2F42 SCS_WAITER_LOOP:
          5A  6A D0 2F42        MOVL    (R10),R10                  ; Get forward link to next entry in the
                    2F45                                           ;  SCS wait queue.
          57  6A D1 2F45        CMPL    (R10),R7                   ; Does the forward link point back to
                    2F48                                           ;  the first forward link?
          04  13    2F48        BEQL    SCS_WT_LP_END              ; If so, no more SCS waiters.
          50  D6    2F4A        INCL    R0                         ; Increase count of SCS waiters.
          F4  11    2F4C        BRB     SCS_WAITER_LOOP            ; Loop for all SCS waiters.
                    2F4E
                    2F4E SCS_WT_LP_END:
       38 A9  50 D0 2F4E        MOVL    R0,PB$L_WAITQFL(R9)        ; Store the number of SCS waiter at
                    2F52                                           ;  offset PB$L_WAITQFL in the PB.
```

```
          30  90  2F52        MOVB    #^A/0/,-                ; Fix the unit number in the port name
       27 A9      2F54                PB$T_LPORT_NAME+3(R9)   ;  to be "0" instead of NULL.
                  2F56
     0480 8F  BA  2F56        POPR    #^M<R7,R10>
                  2F5A
                  2F5A
                  2F5A ;
                  2F5A ; This routine is used to count the number of connections on a circuit and if
                  2F5A ; the display of connections is enabled, to store the connection descriptors in
                  2F5A ; CDT buffer.  All the CDT's for connections established over a particular path
                  2F5A ; are linked together and the first is linked to the associated path block.
                  2F5A ; The message associated with the CDT are also stored in CDT buffer immediately
                  2F5A ; following the CDT.  The number of connections on the circuit is stored in
                  2F5A ; the path block for the associated circuit at offset PB$SHC$L_NUM_PB.
                  2F5A ;
                  2F5A ; NOTE: This code does not check for connections that are in the LISTEN state.
                  2F5A ;       See LISTEN_CDT_SNAP:.
                  2F5A ;
                  2F5A ; INPUTS:
                  2F5A ;
                  2F5A ;       R9  -   Address of path block last stored in PB buffer.
                  2F5A ;       R10 -   Address of path block in SCS database for the last path block
                  2F5A ;               stored in PB buffer.
                  2F5A ;
                  2F5A ; REGISTER USAGE:
                  2F5A ;
                  2F5A ;       R3  -   Pointer into CDT buffer where the CDT will be stored.
                  2F5A ;       R4  -   Counter for copying or zeroing message.
                  2F5A ;       R5  -   Pointer to message when copying message into CDT buffer.
                  2F5A ;       R8  -   Used to count the number of connections on a given path.
                  2F5A ;       R10 -   Pointer to CDT in the SCS database.
                  2F5A ;
                  2F5A
                  2F5A ASSUME  SCS$T_SRC_PROC  EQ SCS$T_DST_PROC+16
                  2F5A ASSUME  SCS$B_CON_DAT   EQ SCS$T_SRC_PROC+16
                  2F5A
                  2F5A CDT_SNAP:
     0408 8F  BB  2F5A        PUSHR   #^M<R3,R10>
          58  D4  2F5E        CLRL    R8                      ; Clear counter for number of
                  2F60                                        ;  connections.
53 00000023'EF D0 2F60        MOVL    CDT_BUF_PNTR,R3         ; Get address of first free slot in
                  2F67                                        ;  CDT buffer.
   5A  34 AA  DE  2F67        MOVAL   PB$L_CDTLST(R10),R10    ; Get the address of the first forward
                  2F6B                                        ;  link for the CDT's associated with
                  2F6B                                        ;  this path block.
   5A     6A  D0  2F6B        MOVL    (R10),R10               ; Are there any CDTs on this path
                  2F6E                                        ;  block?
                  2F6E 5$:    JEQL    CDT_SNAP_DONE           ; If not go on to next path block.
          58  D6  2F73        INCL    R8                      ; Increase count of connections on this
                  2F75                                        ;  circuit.
 00000060 8F  D3  2F75        BITL    #CLS$M_CNT!CLS$M_CRD,-  ; Is the display of counters or credits
 00000001'EF      2F7B                CLASS_SELECT            ;  enabled?
          20  12  2F80        BNEQ    20$                     ; Branch if so.
          04  E1  2F82        BBC     #CLS$V_CON,-            ; If the display of connections is also
 1E 00000001'EF   2F84                CLASS_SELECT,30$        ;  disabled, then skip storing of CDT.
       28 AA  B1  2F8A        CMPW    CDT$W_STATE(R10),-      ; Is the connection in the open state?
          02      2F8D                #CDT$C_OPEN             ;
```

```
                        0A  12  2F8E            BNEQU   10$                             ; Branch if not.
                        0B  E0  2F90            BBS     #CLS$V_OPEN_CON,-               ; Branch to link first CDT if we're
          0A 00000001'EF        2F92                    CLASS_SELECT,20$                ;  displaying open connections.
                        0E  11  2F98            BRB     30$                             ; Otherwise skip first link.
                        0C  E1  2F9A  10$:      BBC     #CLS$V_NOOPEN_CON,-             ; Branch if we're not displaying
          06 00000001'EF        2F9C                    CLASS_SELECT,30$                ;  connections in any other state.
               34 A9    53  D0  2FA2  20$:      MOVL    R3,PB$L_CDTLST(R9)              ; Else, link the first CDT to the PB.
                        3D  11  2FA6            BRB     CDT_FIRST_LOOP                  ; Don't link to previous CDT.
                                2FA8
              5A    6C  AA  D0  2FA8  30$:      MOVL    CDT$L_CDTLST(R10),R10           ; Get the forward link to the next CDT.
                        C0  11  2FAC            BRB     5$                              ; And loop.
                                2FAE
                                2FAE  CDT_LOOP:
              00000041'EF   B7  2FAE            DECW    CDT_DSCR+2                      ; Any room left in connections buffer?
                        53  13  2FB4            BEQLU   CDT_SNAP_DONE                   ; Branch if not.
              5A    6C  AA  D0  2FB6            MOVL    CDT$L_CDTLST(R10),R10           ; Get forward link to CDT.
                        4D  13  2FBA            BEQL    CDT_SNAP_DONE                   ; Branch if there are no (more) CDT's.
                    28  AA  B1  2FBC            CMPW    CDT$W_STATE(R10),-              ; Is the connection in the open state?
                        02      2FBF                    #CDT$C_OPEN                     ;
                        11  12  2FC0            BNEQU   10$                             ; Branch if not.
                        58  D6  2FC2            INCL    R8                              ; Increase count of connections on this
                                2FC4                                                    ;  circuit.
                            D3  2FC4            BITL    #<CLS$M_OPEN_CON!-              ; Are we displaying open connections,
                                2FC5                    CLS$M_CNT!CLS$M_CRD>,-          ;  counters or credits?
  00000001'EF    00000860   8F  2FC5                    CLASS_SELECT                    ;
                        0F  12  2FCF            BNEQ    20$                             ; Branch if so.
                        DB  11  2FD1            BRB     CDT_LOOP                        ; Otherwise, don't link CDT.
                            D3  2FD3  10$:      BITL    #<CLS$M_NOOPEN_CON!-            ; Are we displaying connections that
                                2FD4                    CLS$M_CRD!CLS$M_CRD>,-          ;  aren't open, counters, or credits?
  00000001'EF    00001040   8F  2FD4                    CLASS_SELECT                    ;
                        CE  13  2FDE            BEQL    CDT_LOOP                        ; If not, don't link CDT.
            FF54 C3     53  D0  2FE0  20$:      MOVL    R3,-                            ; Link previous CDT to this one.
                                2FE5                    -<SHC$C_CDT_LENGTH+32>(R3)      ; Total length of CDT is number of
                                2FE5                                                    ;  bytes stored + 32 byte message.
                                2FE5  CDT_FIRST_LOOP:
                        83  D4  2FE5            CLRL    (R3)+                           ; Assume this is the last CDT;
                                2FE7                                                    ;  Zero forward link.
                  0088 8F   28  2FE7            MOVC3   #SHC$C_CDT_LENGTH-4,-           ; Save a portion of the CDT in CDT
                    14 AA       2FEB                    SHC$L_CDT_BEGIN(R10),-          ;  buffer.  (Length -4 because forward
                        63      2FED                    (R3)                            ;  link is not copied.)
                                2FEE                                                    ;  Note: R4 is cleared.
              55    2C  AA  D0  2FEE            MOVL    CDT$L_SCSMSG(R10),R5            ; Is there a message associated with
                                2FF2                                                    ;  this CDT?
                        0D  13  2FF2            BEQL    NO_MESSAGE                      ; Branch if not.
              55    04  A5  DE  2FF4            MOVAL   SCS$T_DST_PROC(R5),R5           ; Get the address of the process names
                                2FF8                                                    ;  in the message.
                  83    85  7D  2FF8  MESSAGE:MOVQ    (R5)+,(R3)+                     ; Move message 8 bytes at a time.
               F9 54    04  F2  2FFB            AOBLSS  #4,R4,MESSAGE                   ; Loop for 32 bytes.
                        AD  11  2FFF            BRB     CDT_LOOP                        ; Loop for all CDT's.
                                3001
                                3001  NO_MESSAGE:
                        83  7C  3001            CLRQ    (R3)+                           ; Zero out the messages area.
               FA 54    04  F2  3003            AOBLSS  #4,R4,NO_MESSAGE                ; Loop for 32 bytes.
                        A5  11  3007            BRB     CDT_LOOP                        ; Loop for all CDT's.
                                3009
                                3009  CDT_SNAP_DONE:
               04 A9    58  D0  3009            MOVL    R8,PB$SHC$L_NUM_CDT(R9) ; Save number of connections on this
```

```
                          300D                                                        ;  path in the path block.
00000023'EF    53    D0   300D                MOVL    R3,CDT_BUF_PNTR                ; Save address of next free slot in
                          3014                                                        ;  CDT buffer.
        0408 8F    BA     3014                POPR    #^M<R3,R10>
                          3018
                          3018  CDT_SNAP_END:
          FEF0    31      3018                BRW     PB_LOOP                        ; Loop for all path blocks on this
                          301B                                                        ;  system block.
                          301B
                          301B  PB_LOOP_END:
0000001F'EF    53    D0   301B                MOVL    R3,PB_BUF_PNTR                 ; Save address of next free entry in
                          3022                                                        ;  PB buffer.
        0688 8F    BA     3022                POPR    #^M<R3,R7,R9,R10>
                          3026
          FE3B    31      3026                BRW     SB_LOOP                        ; Loop for all system blocks.
                          3029
                          3029  SB_LOOP_END:
                          3029
00000017'EF    53    D0   3029                MOVL    R3,SB_BUF_PNTR                 ; Save address of next free entry in
                          3030                                                        ;  SB buffer.
                          3030
                          3030  ;
                          3030  ; This routine executes if the display of circuits in states other than open
                          3030  ; is enabled.  It scans the SCS directory and checks for local processes with
                          3030  ; connections in the LISTEN state.  If any listening CDT's are found, they are
                          3030  ; copied into CDT buffer and linked together.  The first CDT is linked to a
                          3030  ; dummy path block which is in turn linked to the system block for the local
                          3030  ; system.  The local process name is copied from the SCS directory entry and
                          3030  ; stored with the CDT.  The remote process name is zero out.
                          3030  ;
                          3030  ; In the event that there is no system block for the local system block, the
                          3030  ; dummy path block is linked to the dummy system block.
                          3030  ;
                          3030  ; REGISTER USAGE:
                          3030  ;
                          3030  ;       R3  -   Pointer into CDT buffer where the CDT will be stored.
                          3030  ;       R4  -   Used to determine the address where the link to the first
                          3030  ;               listening CDT should be stored.  This address will either be
                          3030  ;               in the dummy path block or in the last path block on the local
                          3030  ;               system system block.
                          3030  ;       R5  -   Used to copy the local process name from the directory entry to
                          3030  ;               CDT buffer.
                          3030  ;       R6  -   Pointer to SCS connection descriptor list.
                          3030  ;       R7  -   Pointer to entry in SCS directory.
                          3030  ;       R9  -   Pointer to dummy path block.
                          3030  ;       R10 -   Used to convert index from the SCS directory to the address of
                          3030  ;               the CDT.
                          3030  ;
                          3030  ;
                          3030
                          3030  LISTENER_SNAP:
            0C    E0      3030                BBS     #CLS$V_NOOPEN_CON,-            ; Execute this code if display of
   03 00000001'EF         3032                        CLASS_SELECT,5$                ;  connections in states other than
                          3038                                                        ;  open is enabled.
          0099    31      3038                BRW     COMMON_SNAP                    ; Other wise jump to common code.
53    00000023'EF    D0   303B  5$:           MOVL    CDT_BUF_PNTR,R3                ; Get address to store next CDT in
                          3042                                                        ;  CDT buffer.
59    00000253'EF    DE   3042                MOVAL   DUMMY_PB,R9                    ; Get address of dummy path block.
```

```
          34 A9  D4  3049                CLRL    PB$L_CDTLST(R9)           ; Assume there are no listeners.
57  00000000'GF  DE  304C                MOVAL   G^SCS$GQ_DIRECT,R7        ; Get address of first SCS directory
                     3053                                                  ;  forward link.
56  00000000'GF  D0  3053                MOVL    G^SCS$GL_CDL,R6           ; Get the address of the SCS connection
                     305A                                                  ;  descriptor list.
                     305A
                     305A   SCS_DIR_LOOP:
    00000041'EF  B5  305A                TSTW    CDT_DSCR+2                ; Is there any room on the CDT buffer?
             6B  13  3060                BEQLU   LISTENER_SNAP_END         ; Branch if not.
         57  67  D0  3062                MOVL    (R7),R7                   ; Get the forward link to first SCS
                     3065                                                  ;  directory entry.
 00000000'8F 57  D1  3065                CMPL    R7,#SCS$GQ_DIRECT         ; Does the link point back to the
                     306C                                                  ;  address of the first link?
             5F  13  306C                BEQL    LISTENER_SNAP_END         ; If so, no more directory entries.
     5A   2C A7  3C  306E                MOVZWL  SDIR$L_CONID(R7),R10      ; Get index value for this CDT.
     5A    664A  D0  3072                MOVL    (R6)[R10],R10             ; Get address of the CDT.
          28 AA  B1  3076                CMPW    CDT$W_STATE(R10),-        ; Is this CDT in the ''listen'' state?
             01      3079                        #CDT$C_LISTEN             ;
             DE  12  307A                BNEQ    SCS_DIR_LOOP              ; If not, look at next directory entry.
          34 A9  D5  307C                TSTL    PB$L_CDTLST(R9)           ; Have we already linked the first
                     307F                                                  ;  listening CDT to the dummy PB?
             28  12  307F                BNEQ    30$                       ; Branch if we have.
     34 A9   53  D0  3081                MOVL    R3,PB$L_CDTLST(R9)        ; Otherwise, link it now.
54  00000013'EF  D0  3085                MOVL    LOC_SB_ADRS,R4            ; Get the address of the system block
                     308C                                                  ;  for the local system.
             07  12  308C                BNEQ    5$                        ; Branch if there is a system block for
                     308E                                                  ;  the local system.
54  000001F3'EF  DE  308E                MOVAL   DUMMY_SB,R4               ; Otherwise, get the address of the
                     3095                                                  ;  dummy system block.
    54    0C A4  DE  3095   5$:          MOVAL   SB$L_PBFL(R4),R4          ; Get the address of the link to the
                     3099                                                  ;  first path block on this system
                     3099                                                  ;  block.
             64  D5  3099                TSTL    (R4)                      ; Are there any path blocks on this
                     309B                                                  ;  system block?
             07  13  309B                BEQL    20$                       ; If not, link the dummy path block
                     309D                                                  ;  directly to the system block.
         54  64  D0  309D   10$:         MOVL    (R4),R4                   ; Get the link to the next path block.
             64  D5  30A0                TSTL    (R4)                      ; Is this the last path block?
             F9  12  30A2                BNEQ    10$                       ; Loop of not.
         64  59  D0  30A4   20$:         MOVL    R9,(R4)                   ; Link the dummy PB to the last path
                     30A7                                                  ;  block on this system block.
             05  11  30A7                BRB     40$                       ; Don't link listening CDT to any
                     30A9                                                  ;  previous CDT's.
   FF54 C3   53  D0  30A9   30$:         MOVL    R3,-                      ; Link previous CDT to this one.
                     30AE                        -<SHC$C_CDT_LENGTH+32>(R3)  ; Total length of CDT is number of
                     30AE                                                  ;  bytes stored + 32 byte message.
             83  D4  30AE   40$:         CLRL    (R3)+                     ; Assume this is the last CDT in the
                     30B0                                                  ;  listen state; Zero forward link.
        0088 8F  28  30B0                MOVC3   #SHC$C_CDT_LENGTH-4,-     ; Save a portion of the CDT in CDT
          14 AA      30B4                        SHC$L_CDT_BEGIN(R10),-    ;  buffer.  (Length -4 because forward
             63      30B6                        (R3)                      ;  link is not copied.)
                     30B7                                                  ;  Note: R4 is cleared.
    55    0C A7  DE  30B7                MOVAL   SDIR$B_PROCNAM(R7),R5     ; Get address of local process name in
                     30BB                                                  ;  directory entry.
         83  85  7D  30BB                MOVQ    (R5)+,(R3)+               ; Save local process name immediately
         83  85  7D  30BE                MOVQ    (R5)+,(R3)+               ;  following the CDT.
             83  7C  30C1                CLRQ    (R3)+                     ; Zero out remote process name.
```

```
              83    7C  30C3         CLRQ    (R3)+                    ; All 16 bytes...
     00000041'EF    B7  30C5         DECW    CDT_DSCR+2               ; Count down the number of entries that
                        30CB                                          ;  will fit in the CDT buffer.
              8D    11  30CB         BRB     SCS_DIR_LOOP             ; Loop for all directory entries.
                        30CD
                        30CD LISTENER_SNAP_END:
  00000023'EF  53   D0  30CD         MOVL    R3,CDT_BUF_PNTR          ; Save address to store next CDT.
                        30D4
                        30D4 COMMON_SNAP:
              0A    E0  30D4         BBS     #CLS$V_NOOPEN_CIR,-      ; Continue with the following code
     00000001'EF        30D6                 CLASS_SELECT,-           ;   if we're displaying formative paths.
              03        30DB                 LOCAL_SNAP               ;
            01F2    31  30DC         BRW     SNAPSHOT_END             ; Otherwise finish up.
                        30DF
                        30DF ;
                        30DF ; This code is always executed for the local report.  It is also executed for
                        30DF ; the cluster report if circuits in states other than open are being displayed.
                        30DF ; It scans the device list looking for PAA0 or PAB0 to determine what ports are
                        30DF ; on the local system so that the associated DDBs, UCBs, and PDTs can be
                        30DF ; located.
                        30DF ;
                        30DF ; REGISTER USAGE:
                        30DF ;
                        30DF ;       R3  -   Address in PDT_BUF where the PDT will be stored.
                        30DF ;       R8  -   Pointer into MISC_BUF where miscellaneous information for the
                        30DF ;               port will be stored.  Data for PAA0 starts at MISC_BUF.  Data
                        30DF ;               for PAB0 starts at MISC_BUF+MISC_BUF_LEN.
                        30DF ;       R10 -   Used as pointer to DDB's.
                        30DF ;
                        30DF
                        30DF LOCAL_SNAP:
 53  000002A7'EF    DE  30DF         MOVAL   PDT_BUF,R3               ; Get address of PDT buffer.
 5A  00000000'GF    D0  30E6         MOVL    G^IOC$GL_DEVLIST,R10     ; Get address of device list.
 58  00000419'EF    DE  30ED         MOVAL   MISC_BUF-MISC_BUF_LEN,R8; Initialize pointer to MISC_BUF.
                        30F4
                        30F4 DDB_LOOP:
          5A   6A   D0  30F4         MOVL    (R10),R10                ; Get address of DDB.
               03   12  30F7         BNEQ    10$                      ; Continue if more DDB's.
             01D5   31  30F9         BRW     SNAPSHOT_END             ; Otherwise, done if no more DDBs.
41415003 8F  14 AA  D1  30FC 10$:    CMPL    DDB$T_NAME(R10),#PAAC    ; Is this DDB for PAA0?
               0A   13  3104         BEQL    20$                      ; Branch if so.
42415003 8F  14 AA  D1  3106         CMPL    DDB$T_NAME(R10),#PABC    ; Is this DDB for PAB0?
               E4   12  310E         BNEQ    DDB_LOOP                 ; If not continue looking.
          58   2E   C0  3110 20$:    ADDL    #MISC_BUF_LEN,R8         ; Get address to store data for PA%0 in
                        3113                                          ;  miscellaneous buffer.
              00    E0  3113         BBS     #RPT$V_CLSTR,-           ; Branch to the formative path block
     00000000'EF        3115                 CURRENT_REPORT,-         ; snapshot routine if we are displaying
              03        311A                 FORM_PB_SNAP             ; the cluster report.
            00BB    31  311B         BRW     PORT_SNAP                ; Otherwise, branch to the port
                        311E                                          ;  snapshot routine.
                        311E
                        311E ;
                        311E ; This code executes only if the cluster report is selected and display of
                        311E ; circuits in states other than open is enabled.  This routine stores formative
                        311E ; path blocks in PB buffer.  It checks for a formative system block associated
                        311E ; with the formative path block.
                        311E ;
```

```
                          311E  ; If there is no formative system block associated with the formative path
                          311E  ; block then it links the formative path block to a dummy system block.
                          311E  ;
                          311E  ; If the formative path block does have a formative system block associated it
                          311E  ; checks if the system ID of the formative system block matchs that of any of
                          311E  ; the system blocks stored in SB buffer.  If there is no match, then the
                          311E  ; formative system block is stored in SB buffer and the formative path block is
                          311E  ; linked to it.
                          311E  ;
                          311E  ; If there is a match, then the incarnation number of the formative system
                          311E  ; block is checked against the incarnation number of the system block stored in
                          311E  ; SB buffer.  If the incarnation numbers are the same, then the formative path
                          311E  ; block is linked to the system block in SB buffer.  If the incarnation numbers
                          311E  ; are different, then the formative system block is stored in SB buffer and the
                          311E  ; formative path block is linked to it.
                          311E  ;
                          311E  ; INPUTS:
                          311E  ;
                          311E  ;         R10 -   Points to base address of DDB.
                          311E  ;
                          311E  ; REGISTER USAGE:
                          311E  ;
                          311E  ;         R3  -   Used to scan the system blocks in SB buffer for a system whose
                          311E  ;                 ID matches the formative system block.  If found points to
                          311E  ;                 the system block.  If not found points to dummy system block.
                          311E  ;             -   Used to find the address last formative path block on the
                          311E  ;                 formative system block or the dummy system block.
                          311E  ;             -   Used as pointer into PB buffer where the formative path block
                          311E  ;                 is to be stored.
                          311E  ;         R7  -   Address of forward link to first formative path block on the
                          311E  ;                 PDT.
                          311E  ;         R9  -   Address of formative system block if available.
                          311E  ;         R10 -   Used to get address of UCB from DDB and address of PDT from
                          311E  ;                 UCB.
                          311E  ;             -   Pointer to formative path block.
                          311E  ;
                          311E  ;
                          311E  ;
                          311E  FORM_PB_SNAP:
               0A   E1    311E          BBC     #CLS$V_NOOPEN_CIR,-         ; If display of circuits in states
  CE 00000001'EF          3120                  CLASS_SELECT,DDB_LOOP       ;  other than open is disabled then
                          3126                                              ;  continue scanning for DDB's.
          070C 8F   BB    3126          PUSHR   #^M<R2,R3,R8,R9,R10>
      5A    04 AA   D0    312A          MOVL    DDB$L_UCB(R10),R10          ; Get address of UCB0 for this port.
  5A      0084 CA   D0    312E          MOVL    UCB$L_PDT(R10),R10          ; Get address of PDT for this port.
                          3133          JEQL    FORM_PB_DONE                ; Branch if no PDT.
  57      0174 CA   DE    3138          MOVAL   PDT$Q_FORMPB(R10),R7        ; Get address of the forward link for
                          313D                                              ;  the first formative path block on
                          313D                                              ;  this PDT.
            5A  57  D0    313D          MOVL    R7,R10                      ; Get copy in R10.
                          3140  FORM_PB_LOOP:
            5A  6A  D0    3140          MOVL    (R10),R10                   ; Get address of formative path block.
            57  5A  D1    3143          CMPL    R10,R7                      ; Does the forward link point to the
                          3146                                              ;  address of the first link?
                          3146          JEQL    FORM_PB_DONE                ; If so, no (more) formative path
                          314B                                              ;  blocks.  Go check for another port.
      59    30 AA   D0    314B          MOVL    PB$L_SBLINK(R10),R9         ; Get formative system block link.
```

```
             4A   13  314F        BEQL    40$                      ; Branch if there is no system block
                      3151                                         ;   associated with this path block.
58    00000047'EF D0  3151        MOVL    VAR_BUF_ADRS,R8          ; Point to the system block buffer.
          18 A8   D1  3158  10$:  CMPL    SB$B_SYSTEMID(R8),-      ; Does the system ID of the formative
          18 A9       315B                SB$B_SYSTEMID(R9)        ;   system block match the ID of any of
                      315D                                         ;   the system blocks stored in SB
                      315D                                         ;   buffer?
             15   12  315D        BNEQ    20$                      ; Branch if not.
          1C A8   B1  315F        CMPW    SB$B_SYSTEMID+4(R8),-    ; Check all six bytes.
          1C A9       3162                SB$B_SYSTEMID+4(R9)      ;
             0E   12  3164        BNEQ    20$                      ; Branch if no match.
          2C A8   D1  3166        CMPL    SB$Q_SWINCARN(R8),-      ; Does the boot time in the formative
          2C A9       3169                SB$Q_SWINCARN(R9)        ;   system block match the system block
                      316B                                         ;   we've found?
             0C   12  316B        BNEQ    30$                      ; Branch if not.
          30 A8   D1  316D        CMPL    SB$Q_SWINCARN+4(R8),-    ; Check the full boot time.
          30 A9       3170                SB$Q_SWINCARN+4(R9)      ;
             2E   13  3172        BEQL    50$                      ; Branch if the boot time matches.
                      3174
       58    68   D0  3174  20$:  MOVL    SB$L_FLINK(R8),R8        ; Get address of next system block.
             DF   12  3177        BNEQ    10$                      ; Loop for all system blocks in SB
                      3179                                         ;   buffer.
                      3179
                      3179  ;
                      3179  ; Save the formative system block if it's ID doesn't match any of the system
                      3179  ; blocks in SB buffer or if the ID matches, but the incarnation number doesn't.
                      3179  ;
                      3179
53    00000017'EF D0  3179  30$:  MOVL    SB_BUF_PNTR,R3           ; Get the address in SB buffer to store
                      3180                                         ;   system block.
      A0 A3  53   D0  3180        MOVL    R3,-SB$C_LENGTH(R3)      ; Link previous system block to this
                      3184                                         ;   one.
       58    53   D0  3184        MOVL    R3,R8                    ; Save the address where this system
                      3187                                         ;   block will be stored.
63   69   0060 8F 28  3187        MOVC3   #SB$C_LENGTH,(R9),(R3)   ; Save the system block in SB buffer.
00000017'EF  53   D0  318D        MOVL    R3,SB_BUF_PNTR           ; Save address to store next system
                      3194                                         ;   block.
             68   D4  3194        CLRL    (R8)                     ; Indicate that this is the last system
                      3196                                         ;   block.
          0C A8   D4  3196        CLRL    SB$L_PBFL(R8)            ; Indicate that there are no path
                      3199                                         ;   blocks on this system block.
             07   11  3199        BRB     50$                      ; Jump to common code.
58    000001F3'EF DE  319B  40$:  MOVAL   DUMMY_SB,R8              ; Get the address of the fake system
                      31A2                                         ;   block.
     58   0C A8   DE  31A2  50$:  MOVAL   SB$L_PBFL(R8),R8         ; Get address of first forward link to
                      31A6                                         ;   the path blocks on this system
                      31A6                                         ;   block.
             68   D5  31A6  60$:  TSTL    (R8)                     ; Is this the last formative path
                      31A8                                         ;   block?
             05   13  31A8        BEQL    70$                      ; Branch if so.
       58    68   D0  31AA        MOVL    (R8),R8                  ; Get link to next formative PB.
             F7   11  31AD        BRB     60$                      ; Loop till last FORM_PB is found.
68    0000001F'EF D0  31AF  70$:  MOVL    PB_BUF_PNTR,(R8)         ; Link the new formative path block
                      31B6                                         ;   to the last one.
       53    68   D0  31B6        MOVL    (R8),R3                  ; Get address where the formative PB
                      31B9                                         ;   will be stored.
63   6A   0054 8F 28  31B9        MOVC3   #PB$C_LENGTH,(R10),(R3)  ; Save the formative PB in PB buffer.
```

```
          AC A3    D4  31BF        CLRL    -PB$C_LENGTH(R3)             ; Indicate that the formative PB is
                       31C2                                             ;  now the last path block on this
                       31C2                                             ;  system block.
          E0 A3    D4  31C2        CLRL    -PB$C_LENGTH+PB$L_CDTLST(R3)
                       31C5                                             ; Zero the link to any connections.
          B0 A3    D4  31C5        CLRL    -PB$C_LENGTH+PB$SHC$L_NUM_CDT(R3)
                       31C8                                             ; Zero the number of connections on the
                       31C8                                             ;  formative path block.
 0000001F'EF    53 D0  31C8        MOVL    R3,PB_BUF_PNTR               ; Save the address to store the next
                       31CF                                             ;  PB.
          FF6E     31  31CF        BRW     FORM_PB_LOOP                 ; Loop for all formative path blocks.
                       31D2
                       31D2 FORM_PB_DONE:
       070C 8F     BA  31D2        POPR    #^M<R2,R3,R8,R9,R10>
         FF1B      31  31D6        BRW     DDB_LOOP                     ; Go check for another port.
                       31D9
                       31D9 ;
                       31D9 ; This code executes only if the local port report in selected.  It stores
                       31D9 ; miscellaneous data about the port and data from the interlocked queues in
                       31D9 ; MISC_BUF.  It stores the PDT's in PDT_BUF.
                       31D9 ;
                       31D9 ; INPUTS:
                       31D9 ;
                       31D9 ;         R10 -   Points to base address of DDB.
                       31D9 ;         R8  -   Pointer into MISC_BUF where miscellaneous information for the
                       31D9 ;                 port will be stored.
                       31D9 ;
                       31D9 ; REGISTER USAGE:
                       31D9 ;
                       31D9 ;         R7  -   Used to get address of UCB from DDB and address of PDT from
                       31D9 ;                 UCB.
                       31D9 ;         R8  -   Pointer to MISC_BUF.
                       31D9 ;         R10 -   Pointer to DDB.
                       31D9 ;
                       31D9
                       31D9 PORT_SNAP:
                       31D9
                       31D9 ;
                       31D9 ; Convert the port name for this port from "PA%" to "PA%0" and save in
                       31D9 ; MISC_BUF.
                       31D9 ;
  14 AA    F8 8F   78  31D9        ASHL    #-8,DDB$T_NAME(R10),-        ; Get the port name shifted left one
              68       31DE                MB$L_PORT_NAME(R8)           ;  byte.
   30000000 8F     C8  31DF        BISL    #^X30000000,-                ; AND in a zero at the end of the name.
              68       31E5                MB$L_PORT_NAME(R8)
           14 A8   D4  31E6        CLRL    MB$L_FORM_PB(R8)             ; Initialize count of formative path
                       31E9                                             ;  blocks on this port. (In MISC_BUF)
     57    04 AA   D0  31E9        MOVL    DDB$L_UCB(R10),R7            ; Get address of UCB0 for this port.
           41 A7   9A  31ED        MOVZBL  UCB$B_DEVTYPE(R7),-          ; Get the device type for this port
           08 A8       31F0                MB$L_DEVTYPE(R8)             ;  from UCB0 and save in MISC_BUF.
           64 A7   3C  31F2        MOVZWL  UCB$W_STS(R7),-              ; Get device status for this port
           0C A8       31F5                MB$L_STS(R8)                 ;  from UCB0 and save in MISC_BUF.
         0080 C7   D0  31F7        MOVL    UCB$B_ERTCNT(R7),-           ; Get current retry count, maximum
           10 A8       31FB                MB$B_ERTCNT(R8)              ;  retry count, and error count from
                       31FD                                             ;  UCB0 and save in MISC_BUF.
     57  0084 C7   D0  31FD        MOVL    UCB$L_PDT(R7),R7             ; Get address of PDT for this port.
                       3202        JEQL    FREE_BUF_SNAP                ; Branch if no PDT.
```

```
                            3207
                            3207  ;
                            3207  ; Count the number of formative circuits and save in MISC_BUF.
                            3207  ; Save the port number in MISC_BUF.
                            3207  ; Copy a portion of the PDT in PDT_BUF.
                            3207  ; Count the number of pool waiters and store in MISC_BUF.
                            3207  ;
                            3207  ; INPUTS:
                            3207  ;
                            3207  ;       R3  -   Pointer into PDT_BUF where PDT is to be stored.
                            3207  ;       R7  -   Base address of PDT.
                            3207  ;       R8  -   Base address to store data for this port in MISC_BUF.
                            3207  ;
                            3207  ; REGISTER USAGE:
                            3207  ;
                            3207  ;       R5  -   Pointer to formative path blocks.
                            3207  ;       R5  -   Pointer to entries in pool waiter queue.
                            3207  ;       R6  -   Address of link to first formative path block on the PDT.
                            3207  ;           -   Address of loopback data gram buffer.
                            3207  ;           -   Pointer to first address in PDT to be saved in PDT_BUF.
                            3207  ;           -   Address of link to first pool waiter queue entry.
                            3207  ;       R8  -   Pointer to MISC_BUF.
                            3207  ;
   56    0174 C7    DE      3207          MOVAL   PDT$Q_FORMPB(R7),R6        ; Get address of the forward link for
                            320C                                             ;  the first formative path block on
                            320C                                             ;  this PDT.
         55   56    D0      320C          MOVL    R6,R5                      ; Get copy in R5.
                            320F
                            320F  PORT_FORM_PB_LOOP:
         55   65    D0      320F          MOVL    (R5),R5                    ; Get address of formative path block.
         56   55    D1      3212          CMPL    R5,R6                      ; Does the forward link point to the
                            3215                                             ;  address of the first link?
              05    13      3215          BEQL    PORT_FORM_PB_LOOP_END      ; If so, no (more) formative path
                            3217                                             ;  blocks.  Go check for another port.
         14 A8      D6      3217          INCL    MB$L_FORM_PB(R8)           ; Increase count of formative circuits.
              F3    11      321A          BRB     PORT_FORM_PB_LOOP          ; Loop for all formative paths.
                            321C
                            321C  PORT_FORM_PB_LOOP_END:
   56    0184 C7    D0      321C          MOVL    PDT$L_LBDG(R7),R6          ; Get address of loopback datagram
                            3221                                             ;  buffer.
         0C A6      9A      3221          MOVZBL  PPD$B_PORT(R6),-           ; Save the port number of this port in
         04 A8              3224                  MB$L_PORT_NUM(R8)          ;  MISC_BUF.
   56    00B4 C7    DE      3226          MOVAL   SHC$C_PDT_BEGIN(R7),R6     ; Get address to start copying PDT.
         00D0 8F    28      322B          MOVC3   #SHC$C_PDT_LENGTH,-        ; Get a portion of the PDT into
         63   66            322F                  (R6),(R3)                  ;  PDT_BUF.
   55    00AC C7    DE      3231          MOVAL   PDT$L_WAITQFL(R7),R5       ; Get address of pool waiter queue
                            3236                                             ;  forward link.
         56   55    D0      3236          MOVL    R5,R6                      ; Save a copy in R5.
         18 A8      D4      3239          CLRL    MB$L_PL_WAITERS(R8)        ; Initialize number of pool waiters on
                            323C                                             ;  this port.
                            323C
                            323C  POOL_WAITER_LOOP:
         55   65    D0      323C          MOVL    (R5),R5                    ; Get first forward link.
         56   55    D1      323F          CMPL    R5,R6                      ; Any (more) pool waiters?
              05    13      3242          BEQL    POOL_WAITER_LOOP_END       ; Branch if not.
         18 A8      D6      3244          INCL    MB$L_PL_WAITERS(R8)        ; Increment number of pool waiters on
                            3247                                             ;  this port.
```

```
                F3   11   3247          BRB     POOL_WAITER_LOOP        ; Loop till all pool waiters found.
                          3249
                          3249  POOL_WAITER_LOOP_END:
                          3249
                          3249  ;
                          3249  ; Count the number of entries in the interlocked queue and store in MISC_BUF.
                          3249  ;
                          3249  ; INPUTS:
                          3249  ;
                          3249  ;       R8  -   Pointer into MISC_BUF to store data for the port.
                          3249  ;       R7  -   Pointer to PDT.
                          3249  ;
                          3249  ; REGISTER USAGE:
                          3249  ;
                          3249  ;       R6  -   Pointer to queue to be counted.
                          3249  ;       R7  -   Pointer to PDT.
                          3249  ;       R8  -   Pointer to MISC_BUF.
                          3249  ;       R9  -   Count of entries returned form CHECK_QUE routine.
                          3249  ;
                          3249
                          3249  INTERLOCK_SNAP:
   56    01D0 C7     DE   3249          MOVAL   PDT$Q_DFREEQ(R7),R6     ; Get address of datagram free queue.
                59   D4   324E          CLRL    R9                      ; Initialize count of entries in que.
              00A1   30   3250          BSBW    CHECK_QUE               ; Get the number of entries.
              0082   30   3253          BSBW    CNVRT_ASCII             ; Convert result to ascii string.
     1C A8      59   B0   3256          MOVW    R9,MB$W_DG_FREE(R8)     ; Save number of free datagrams on this
                          325A                                          ;  port.
   56    01D8 C7     DE   325A          MOVAL   PDT$Q_MFREEQ(R7),R6     ; Get the address of the message free
                          325F                                          ;  queue.
                59   D4   325F          CLRL    R9                      ; Initialize count of entries in que.
              0090   30   3261          BSBW    CHECK_QUE               ; Get the number of entries.
                72   10   3264          BSB     CNVRT_ASCII             ; Convert result to ascii string.
     1E A8      59   B0   3266          MOVW    R9,MB$W_MSG_FREE(R8)    ; Save number of free messages on
                          326A                                          ;  this port.
   56    01E0 C7     DE   326A          MOVAL   PDT$Q_COMQL(R7),R6      ; Get address of low priority command
                          326F                                          ;  queue.
                59   D4   326F          CLRL    R9                      ; Initialize count of entries in que.
              0080   30   3271          BSBW    CHECK_QUE               ; Get the number of entries in the low
                          3274                                          ;  priority command queue.
   56    01E8 C7     DE   3274          MOVAL   PDT$Q_COMQH(R7),R6      ; Get the address of the high priority
                          3279                                          ;  command queue.
                79   10   3279          BSB     CHECK_QUE               ; Get the number of entries in the high
                          327B                                          ;  priority command queue and add to
                          327B                                          ;  total number of commands queued.
   56    01F0 C7     DE   327B          MOVAL   PDT$Q_COMQ2(R7),R6      ; Get the address of command queue 2.
                72   10   3280          BSB     CHECK_QUE               ; Get the number of entries in command
                          3282                                          ;  queue 2 and add to total number of
                          3282                                          ;  commands qued.
   56    01F8 C7     DE   3282          MOVAL   PDT$Q_COMQ3(R7),R6      ; Get the address of command queue 3.
                6B   10   3287          BSB     CHECK_QUE               ; Get the number of entries in command
                          3289                                          ;  queue 3 and add to total number of
                          3289                                          ;  commands qued.
                4D   10   3289          BSB     CNVRT_ASCII             ; Convert result to ascii string.
     20 A8      59   B0   328B          MOVW    R9,MB$W_CMDS_QUED(R8)   ; Save the total number of commands
                          328F                                          ;  queue on this port.
   56    0200 C7     DE   328F          MOVAL   PDT$Q_RSPQ(R7),R6       ; Get the address of the response
                          3294                                          ;  queue.
```

```
                 59   D4 3294        CLRL    R9                          ; Initialize count of entries in que.
                 5C   10 3296        BSB     CHECK_QUE                   ; Get the number of entries in the
                         3298                                            ;  response queue.
                 3E   10 3298        BSB     CNVRT_ASCII                 ; Convert result to ascii string.
          22 A8  59   B0 329A        MOVW    R9,MB$W_RSPS_QUED(R8)       ; Save the number of responses queued
                         329E                                            ;  on this port.
                         329E
                         329E  ;
                         329E  ; Count the number of free buffer descriptors and save in NUM_FREE_BUF_DSCR.
                         329E  ; Calculate the number of buffer descriptors in use and save in NUM_BUF_DSCR.
                         329E  ;
                         329E  ; REGISTER USAGE:
                         329E  ;
                         329E  ;     R6  -  Counter for number of free buffer descriptors.
                         329E  ;     R7  -  Pointer to free buffer descriptors.
                         329E  ;
                         329E
                         329E  FREE_BUF_SNAP:
     57   00000000'GF    D0 329E     MOVL    G^SCS$GL_BDT,R7             ; Get the address of the buffer
                         32A5                                            ;  descriptor table header.
     57   FFFFFFE8 8F    C0 32A5     ADDL    #CIBDT$L_FREEBD-CIBD$L_LINK,R7
                         32AC                                            ; Get the link to the first buffer
                         32AC                                            ;  descriptor.
                 56   D4 32AC        CLRL    R6                          ; Assume no free buffer descriptors.
                         32AE
                         32AE  FREE_BUF_LOOP:
          57   0C A7  D0 32AE        MOVL    CIBD$L_LINK(R7),R7          ; Are there any (more) free buffer
                         32B2                                            ;  descriptors?
                 04   13 32B2        BEQL    FREE_BUF_END                ; Branch if not.
                 56   D6 32B4        INCL    R6                          ; Increase count of free buffer
                         32B6                                            ;  descriptors for this port.
                 F6   11 32B6        BRB     FREE_BUF_LOOP               ; Loop for all free buffer descriptors.
                         32B8
                         32B8  FREE_BUF_END:
     0000000F'EF  56  D0 32B8        MOVL    R6,NUM_FREE_BUF_DSCR        ; Save number of free buffer
                         32BF                                            ;  descriptors.
     57   00000000'GF    3C 32BF     MOVZWL  G^SCS$GW_BDTCNT,R7          ; Get the total number of buffer
                         32C6                                            ;  descriptors available.
0000000B'EF   57  56  C3 32C6        SUBL3   R6,R7,NUM_BUF_DSCR          ; Save the number of buffer descriptors
                         32CE                                            ;  in use.
               FE23   31 32CE        BRW     DDB_LOOP                    ; Loop for all ports.
                         32D1
                         32D1  PORT_SNAP_END:
                         32D1
                         32D1  SNAPSHOT_END:
              50  01  D0 32D1        MOVL    #SS$_NORMAL,R0              ; Indicate success.
                         32D4
                         32D4  SNAPSHOT_ERR:
                         32D4        ENBINT                              ; Lower IPL.
                      04 32D7        RET                                 ; And return.
                         32D8
```

```
                        32D8 ;++
                        32D8 ;
                        32D8 ; FUNCTIONAL DESCRIPTION:
                        32D8 ;
                        32D8 ;       This routine converts the number of entries found in an interlock queue
                        32D8 ;       to an ascii string.  If the number of entries found exceeds QUEUE_LIMIT
                        32D8 ;       then a carrot ">" is placed before the count when it is output.
                        32D8 ;
                        32D8 ; CALLING SEQUENCE:
                        32D8 ;
                        32D8 ;       JSB     CNVRT_ASCII
                        32D8 ;
                        32D8 ; CALLED BY:
                        32D8 ;
                        32D8 ;       INTERLOCK_SNAP
                        32D8 ;
                        32D8 ; INPUTS:
                        32D8 ;
                        32D8 ;       Number of entries in the queue in R9.
                        32D8 ;
                        32D8 ; OUTPUTS:
                        32D8 ;
                        32D8 ;       Ascii string returned in R9.
                        32D8 ;
                        32D8 ; RETURN STATUS:
                        32D8 ;
                        32D8 ;       None.
                        32D8 ;--
                        32D8
                        32D8 CNVRT_ASCII:
      59   03   91      32D8         CMPB    #QUE_LIMIT,R9           ; Does the result equal the limit.
           0B   13      32DB         BEQL    10$                     ; Branch if so.
  59  59   08   78      32DD         ASHL    #8,R9,R9                ; Get the count into the second byte.
  59  3020 8F   A0      32E1         ADDW    #^A/ 0/,R9              ; Convert second byte to ascii and make
                        32E6                                         ;  the first byte a blank.
           0B   11      32E6         BRB     20$                     ; Branch to common code.
           59   D7      32E8 10$:    DECL    R9                      ; 3 becomes >2.
  59  59   08   78      32EA         ASHL    #8,R9,R9                ; Get the count into the second byte.
  59  303E 8F   A0      32EE         ADDW    #^A/>0/,R9              ; Convert second byte to ascii and make
                        32F3                                         ;  the first byte a ">"
                05      32F3 20$:    RSB
                        32F4
```

```
                              32F4          .SBTTL          -       Interlock Queue routine.
                              32F4  ;++
                              32F4  ;
                              32F4  ; FUNCTIONAL DESCRIPTION:
                              32F4  ;
                              32F4  ;       This routine is used to count the number of entries in an interlocked
                              32F4  ;        queue.  The number of entries found is returned in R9.  Note that R9
                              32F4  ;        is not cleared by this routine so that R9 can be used to count the
                              32F4  ;        total number of entries in several queues if this routine is called
                              32F4  ;        repeatedly.  Since this routine executes at IPL 31 the maximum number
                              32F4  ;        of entries counted by this routine has been limited to 3.
                              32F4  ;
                              32F4  ; CALLING SEQUENCE:
                              32F4  ;
                              32F4  ;       JSB CHECK_QUE
                              32F4  ;
                              32F4  ; CALLED BY:
                              32F4  ;
                              32F4  ;       INTERLOCK_SNAP
                              32F4  ;
                              32F4  ; INPUTS:
                              32F4  ;
                              32F4  ;       R6  = Address of queue for which entries are to be counted.
                              32F4  ;
                              32F4  ; OUTPUTS:
                              32F4  ;
                              32F4  ;       The number of entries in the queue is added to R9.
                              32F4  ;
                              32F4  ; IMPLICIT OUTPUTS:
                              32F4  ;
                              32F4  ;       R5 scratched.
                              32F4  ;
                              32F4  ; RETURN STATUS:
                              32F4  ;
                              32F4  ;       NORMAL or SS$_INTERLOCK status returned in R0.
                              32F4  ;
                              32F4  ; REGISTER USAGE:
                              32F4  ;
                              32F4  ;       R1 -    Interlock retry counter.
                              32F4  ;       R4 -    Offset to next queue entry.
                              32F4  ;       R5 -    Address of queue entry.
                              32F4  ;       R6 -    Address of first entry in the queue.
                              32F4  ;       R9 -    Counter for number of entries in the queue.
                              32F4  ;
                              32F4  ;--
                              32F4
                              32F4  CHECK_QUE:
     50  0000038C 8F     D0   32F4          MOVL    #SS$_INTERLOCK,R0       ; Assume interlock failure.
                  51     D4   32FB          CLRL    R1                      ; Initialize interlock retry counter.
             55   56     D0   32FD          MOVL    R6,R5                   ; Save copy of queue address in R5.
                              3300          DSBINT                          ; Disable interupts.
        54   65   01     CB   3306          BICL3   #1,(R5),R4              ; Clear interlock bit in first
                              330A                                          ;  self-relative offset.
                              330A
                              330A  WAS_LOCKED:
  02 51  00000000'GF     F2   330A          AOBLSS  G^EXE$GL_LOCKRTRY,R1,-  ; Try to lock again if we still have
                              3312                  TRY_LOCK                ;  time.
```

```
              1B  11  3312         BRB     LOCK_FAIL               ; Otherwise, return with failure.
                      3314
                      3314 TRY_LOCK:
       F2 66  00  E6  3314         BBSSI   #0,(R6),WAS_LOCKED      ; Try to lock the queue.
                      3318
        55  6544  9E  3318 LOOP:   MOVAB   (R5)[R4],R5             ; Get forward link to next entry.
           56  55  D1  331C         CMPL    R5,R6                   ; Does this link point back to the
                      331F                                         ;  first link?
              07  13  331F         BEQL    10$                     ; Branch if it does.
           54  65  D0  3321         MOVL    (R5),R4                 ; Get next relative offset.
       F0 59  03  F2  3324         AOBLSS  #QUE_LIMIT,R9,LOOP      ; Increment number of entries and
                      3328                                         ;  loop if limit not exceeded.
                      3328
       00 66  00  E7  3328 10$:    BBCCI   #0,(R6),LOWERIPL        ; Unlock the queue.
                      332C
                      332C LOWERIPL:
           50  01  D0  332C         MOVL    #SS$_NORMAL,R0          ; Indicate success.
                      332F
                      332F LOCK_FAIL:
                      332F         ENBINT                          ; Enable interupts.
                  05  3332         RSB
                      3333
                      3333 SNAP_END:
                      3333
                      3333
```

```
        000004A3 10405          .PSECT  SNAP_DATA,RD,WRT,NOEXE
            04A3 10406
            04A3 10407 ;
            04A3 10408 ; This buffer has been placed here so that the fixed length snapshot buffers
            04A3 10409 ; can be locked in memory at the same time as the snapshot routines.  The
            04A3 10410 ; program will not assemble if this buffer is placed before the snapshot
            04A3 10411 ; buffers.  (For some reason FIX_BUF_LEN cannot be resolved)
            04A3 10412 ;
            04A3 10413
            04A3 10414 OLD_FIX_BUF:                            ; Copy of fix length buffer from
000008FB    04A3 10415          .BLKB   FIX_BUF_LEN             ;  previous snapshot.  No need to
            08FB 10416                                          ;  lock this buffer.
```

```
                              08FB        .SBTTL     -        GET_NUM_PB
                              08FB  ;++
                              08FB  ;
                              08FB  ; FUNCTIONAL DESCRIPTION:
                              08FB  ;
                              08FB  ;       This routine is used to count the number of circuits on a system.
                              08FB  ;       It uses the information stored in PB buffer by the SNAPSHOT
                              08FB  ;       routine to count the path blocks.  The number of circuits is stored in
                              08FB  ;       the associated system block.
                              08FB  ;
                              08FB  ;       This code is separated from the snapshot routine to minimize time spent
                              08FB  ;       at raised IPL.
                              08FB  ;
                              08FB  ; CALLING SEQUENCE:
                              08FB  ;
                              08FB  ;       JSB GET_NUM_PB
                              08FB  ;
                              08FB  ;
                              08FB  ; IMPLICIT INPUTS:
                              08FB  ;
                              08FB  ;       SNAPSHOT routine has been executed and the snapshot data is stored in
                              08FB  ;       SNAP_BUF.
                              08FB  ;
                              08FB  ; OUTPUTS:
                              08FB  ;
                              08FB  ;       Number of path blocks on each system block stored in the respective
                              08FB  ;       system block at offset SB$SHC$L_NUM_PB.
                              08FB  ;
                              08FB  ; REGISTER USAGE:
                              08FB  ;
                              08FB  ;       R2  -   Counts the number of path blocks.
                              08FB  ;       R10 -   Points to the first longword of a path block in PB buffer.
                              08FB  ;       R11 -   Points to the first longword of a connection descriptor in
                              08FB  ;               CDT buffer.
                              08FB  ;
                              08FB  ;
                              08FB  ;--
                              08FB
                              08FB  GET_NUM_PB:
         0C04 8F     BB       08FB          PUSHR   #^M<R2,R10,R11>
5B   000003ED'EF     D0       08FF          MOVL    VAR_BUF,R11                 ; Get address of first system block.
               52    D4       0906  10$:    CLRL    R2                          ; Clear counter for number of paths on
                              0908                                              ;  this system block.
      5A    0C AB    D0       0908          MOVL    SB$L_PBFL(R11),R10          ; Get address of first path block on
                              090C                                              ;  this system block.
               0D    13       090C          BEQL    40$                         ; Branch if there are no circuits on
                              090E                                              ;  this system.
            12 AA    B1       090E  20$:    CMPW    PB$W_STATE(R10),-           ; Is the circuit in the open state?
               03             0911                  #PB$C_OPEN                  ;
               02    12       0912          BNEQ    30$                         ; If not don't count it.
               52    D6       0914          INCL    R2                          ; Increase count of path blocks on this
                              0916                                              ;  system block.
         5A    6A    D0       0916  30$:    MOVL    (R10),R10                   ; Get address of next path block.
               F3    12       0919          BNEQ    20$                         ; Branch if no more circuits on this
                              091B                                              ;  system.
      04 AB    52    D0       091B  40$:    MOVL    R2,SB$SHC$L_NUM_PB(R11)     ; Save number of paths on the system
                              091F                                              ;  block.
         5B    6B    D0       091F          MOVL    (R11),R11                   ; Get address of next system block.
```

```
          E2  12 0922 10475          BNEQ    10$                     ; Loop for all systems in the cluster.
     0C04 8F  BA 0924 10476          POPR    #^M<R2,R10,R11>
              05 0928 10477          RSB
                 0929 10478
```

```
0929            .SBTTL  Parser State Table.
0929
0929            $INIT_STATE     CMD_STATE,CMD_KEY
0929
0929            $STATE  CMD_START
0929
0929            $TRAN   '!',TPA$_EXIT               ; Ignore comments.
0929
0929            $TRAN   'SHOW',SHOW                 ; Command "SHOW".
0929
0929            $TRAN   'ADD',ADD                   ; Command "ADD".
0929
0929            $TRAN   'REMOVE',REMOVE             ; Command "REMOVE".
0929
0929            $TRAN   'SET',SET                   ; Command "SET".
0929
0929            $TRAN   'HELP',HELP                 ; Command "HELP".
0929
0929            $TRAN   'EXIT',TPA$_EXIT,-          ; Command "EXIT".
0929                    CNTRLC_AST_DEL              ;
0929
0929            $TRAN   'INITIALIZE',TPA$_EXIT,-; Command "INITIALIZE".
0929                    RESET
0929
0929            $TRAN   '@',TPA$_EXIT,EXECUTE       ; Execute procedure command.
0929
0929    ;
0929    ; Translate the SHOW CLUSTER and SHOW LOCAL commands.
0929    ;
0929            $STATE  SHOW
0929
0929            $TRAN   TPA$_EOS,TPA$_EXIT,-        ; Return if end of string.
0929                    REFRESH
0929
0929            $TRAN   'CLUSTER',TPA$_EXIT,-       ; "SHOW CLUSTER".
0929                    SHOW_CLUSTER                ;
0929
0929            $TRAN   'LOCAL_PORTS',-             ; "SHOW LOCAL_PORTS".
0929                    TPA$_EXIT,SHOW_LOCAL        ;
0929
0929
```

```
0929 ;
0929 ; Translate the add "class" or "field-name" commands.
0929 ;
0929        $STATE  ADD
0929
0929        $TRAN   TPA$_EOS,TPA$_EXIT,-     ; Return if end of string.
0929                REFRESH
0929
0929        $TRAN   'CLUSTER',ADD_CLUSTER,- ; "ADD CLUSTER"
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   'SYSTEMS',ADD_SYS,-     ; "ADD SYSTEMS".
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   'MEMBERS',ADD_MEMBERS,- ; "ADD MEMBERS.
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   'CONNECTIONS',ADD_CON,- ; "ADD CONNECTIONS".
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   'CIRCUITS',ADD_CIR,-    ; "ADD CIRCUITS".
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   'COUNTERS',-            ; "ADD COUNTERS".
0929                ADD_COUNTERS,-           ;
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   'CREDITS',ADD_CREDITS,- ; "ADD CREDITS".
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   'LOCAL_PORTS',-         ; "ADD LOCAL_PORTS".
0929                ADD_LOCAL_PORTS,-        ;
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   'ERRORS',ADD_ERRORS,-   ; "ADD ERRORS".
0929                ,,TKN_DSCR               ;
0929
0929        $TRAN   TPA$_SYMBOL,,ADD_FIELD  ; "ADD field-name".
0929
0929        $STATE  ADD_NEXT_CLASS
0929        $TRAN   <','>,ADD               ; Loop if another class was
0929                                         ;  specified.
0929        $TRAN   '!',TPA$_EXIT,REFRESH   ; Allow comments...
0929        $TRAN   TPA$_EOS,TPA$_EXIT,-    ;  or end of string.
0929                REFRESH                  ;
0929        $TRAN   TPA$_LAMBDA,TPA$_FAIL,- ;
0929                REFRESH                  ; Anything else is an error.
0929
```

```
0929 ;
0929 ; ADD CLUSTER
0929 ;

0929         $STATE  ADD_CLUSTER

0929         $TRAN   '/',ADD_CLUSTER_ALL

0929         $TRAN   TPA$_LAMBDA,-
0929                 ADD_NEXT_CLASS,-           ;
0929                 INCLUDE_CLASS,,,-          ;
0929                 CLS$M_CLUB                 ;

0929         $STATE  ADD_CLUSTER_ALL
0929         $TRAN   'ALL',ADD_NEXT_CLASS,-     ; ADD CLUSTER/ALL
0929                 ADD_ALL_FIELDS,,,-         ;
0929                 CLS$M_CLUB                 ;

0929 ;
0929 ; ADD MEMBERS
0929 ;

0929         $STATE  ADD_MEMBERS

0929         $TRAN   '/',ADD_MEMBERS_QUAL

0929         $TRAN   TPA$_LAMBDA,-
0929                 ADD_NEXT_CLASS,-           ;
0929                 INCLUDE_CLASS,,,-          ;
0929                 CLS$M_CSB                  ;

0929         $STATE  ADD_MEMBERS_QUAL
0929         $TRAN   'ALL',ADD_NEXT_CLASS,-     ; ADD MEMBERS/ALL
0929                 ADD_ALL_FIELDS,,,-         ;
0929                 CLS$M_CSB                  ;

0929 ;
0929 ; ADD COUNTERS
0929 ;

0929         $STATE  ADD_COUNTERS

0929         $TRAN   '/',ADD_COUNTERS_QUAL

0929         $TRAN   TPA$_LAMBDA,-
0929                 ADD_NEXT_CLASS,-           ;
0929                 INCLUDE_CLASS,,,-          ;
0929                 CLS$M_CNT                  ;

0929         $STATE  ADD_COUNTERS_QUAL
0929         $TRAN   'ALL',ADD_NEXT_CLASS,-     ; ADD COUNTERS/ALL
0929                 ADD_ALL_FIELDS,,,-         ;
0929                 CLS$M_CNT                  ;


0929 ;
0929 ; ADD CREDITS
```

```
0929  ;
0929
0929          $STATE  ADD_CREDITS
0929
0929          $TRAN   '/',ADD_CREDITS_QUAL
0929
0929          $TRAN   TPA$_LAMBDA,-
0929                  ADD_NEXT_CLASS,-          ;
0929                  INCLUDE_CLASS,,,-         ;
0929                  CLS$M_CRD                 ;
0929
0929          $STATE  ADD_CREDITS_QUAL
0929          $TRAN   'ALL',ADD_NEXT_CLASS,-    ; ADD CREDITS/ALL
0929                  ADD_ALL_FIELDS,,,-        ;
0929                  CLS$M_CRD                 ;
0929
0929
0929  ;
0929  ; ADD ERRORS
0929  ;
0929
0929          $STATE  ADD_ERRORS
0929
0929          $TRAN   '/',ADD_ERRORS_QUAL
0929
0929          $TRAN   TPA$_LAMBDA,-
0929                  ADD_NEXT_CLASS,-          ;
0929                  INCLUDE_CLASS,,,-         ;
0929                  CLS$M_ERR                 ;
0929
0929          $STATE  ADD_ERRORS_QUAL
0929          $TRAN   'ALL',ADD_NEXT_CLASS,-    ; ADD ERRORS/ALL
0929                  ADD_ALL_FIELDS,,,-        ;
0929                  CLS$M_ERR                 ;
0929
0929  ;
0929  ; ADD LOCAL_PORTS
0929  ;
0929
0929          $STATE  ADD_LOCAL_PORTS
0929
0929          $TRAN   '/',ADD_LOCAL_PORTS_QUAL
0929
0929          $TRAN   TPA$_LAMBDA,-
0929                  ADD_NEXT_CLASS,-          ;
0929                  INCLUDE_CLASS,,,-         ;
0929                  CLS$M_LPRT                ;
0929
0929          $STATE  ADD_LOCAL_PORTS_QUAL
0929          $TRAN   'ALL',ADD_NEXT_CLASS,-    ; ADD LOCAL_PORTS/ALL
0929                  ADD_ALL_FIELDS,,,-        ;
0929                  CLS$M_LPRT                ;
0929
0929
```

```
0929 ;
0929 ; ADD SYSTEMS
0929 ;
0929         $STATE  ADD_SYS
0929
0929         $TRAN   '/',ADD_SYS_QUAL          ; Go to ADD_SYS_QUAL state if qualifier
0929                                           ;  is present.
0929
0929         $TRAN   TPA$_LAMBDA,-             ; Otherwise just ADD SYSTEMS.
0929                 ADD_NEXT_CLASS,-          ;
0929                 INCLUDE_CLASS,,,-         ;
0929                 CLS$M_SYS                 ;
0929
0929 ;
0929 ; Process ADD SYSTEMS qualifiers.
0929 ;
0929         $STATE  ADD_SYS_QUAL
0929
0929         $TRAN   'ALL',-                   ; ADD SYSTEMS/ALL
0929                 NEXT_ADD_SYS_QUAL,-       ;
0929                 ADD_ALL_FIELDS,,,-        ;
0929                 CLS$M_SYS                 ;
0929         $TRAN   'ID',ADD_SYS_ID           ; ADD SYSTEM/ID
0929         $TRAN   'TYPE',ADD_SYS_TYPE       ; ADD SYSTEM/TYPE
0929         $TRAN   'NODE',ADD_SYS_NODE       ; ADD SYSTEM/NODE
0929         $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929
0929 ;
0929 ; Loop for multiple qualifiers on the SYSTEMS class.
0929 ;
0929         $STATE NEXT_ADD_SYS_QUAL
0929         $TRAN   '/',ADD_SYS_QUAL
0929         $TRAN   TPA$_LAMBDA,ADD_NEXT_CLASS
0929
0929
```

```
0929  ;
0929  ; Process ADD SYSTEMS/ID=system_id
0929  ;         ADD SYSTEMS/ID=(system_id,...)
0929
0929          $STATE  ADD_SYS_ID
0929          $TRAN   '='
0929          $STATE
0929          $TRAN   '(',ADD_ID_LOOP          ; Go parse a list of ID's
0929          $TRAN   TPA$_LAMBDA,             ; Go parse a single ID
0929          $STATE
0929          $TRAN   !ADD_SYS_ID_STRING,-     ;
0929                  NEXT_ADD_SYS_QUAL        ;
0929
0929          $STATE  ADD_ID_LOOP              ; Parse a list of ID numbers till a ")"
0929                                           ;  is found.
0929          $TRAN   !ADD_SYS_ID_STRING       ;
0929          $STATE
0929          $TRAN   <','>,ADD_ID_LOOP        ; Go parse a single ID
0929          $TRAN   ')',NEXT_ADD_SYS_QUAL    ; Exit loop if ")" found.
0929
0929          $STATE  ADD_SYS_ID_STRING        ; Parse a single ID
0929
0929          $TRAN   TPA$_STRING,,,,TKN_DSCR ; Save address of string.
0929          $STATE
0929          $TRAN   TPA$_LAMBDA,TPA$_EXIT,- ; Call REM_ID to remove system from
0929                  REM_ID                   ;  disabled list.
0929
0929
0929
```

```
0929  ;
0929  ; Process ADD SYSTEMS/TYPE=hw_type
0929  ;         ADD SYSTEMS/TYPE=(hw_type,...)
0929
0929          $STATE  ADD_SYS_TYPE
0929          $TRAN   '='
0929          $STATE
0929          $TRAN   '(',ADD_SYS_TYPE_LOOP   ; Go parse a list of system types.
0929          $TRAN   TPA$_LAMBDA,            ; Go parse a single system type.
0929          $STATE
0929          $TRAN   !ADD_SYS_TYPE_STRING,-;
0929                  NEXT_ADD_SYS_QUAL       ;
0929
0929          $STATE  ADD_SYS_TYPE_LOOP       ; Parse a list of system types till a
0929                                          ;  ")" is found.
0929          $TRAN   !ADD_SYS_TYPE_STRING    ;
0929          $STATE
0929          $TRAN   <','>,ADD_SYS_TYPE_LOOP ; Go parse a single system type.
0929          $TRAN   ')',NEXT_ADD_SYS_QUAL   ; Exit loop if ")" found.
0929
0929
0929          $STATE  ADD_SYS_TYPE_STRING     ; Parse a single system type
0929
0929          $TRAN   'ALL',TPA$_EXIT,-       ; "ADD SYSTEMS/TYPE=ALL" - Add all
0929                  INCLUDE_CLASS,,,-       ; types of systems to the display.
0929                  CLS$M_ALL_SYS
0929
0929          $TRAN   'V780',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=780" - Add only
0929                  INCLUDE_CLASS,,,-       ;   11/780 systems to the display.
0929                  CLS$M_780_SYS           ;
0929
0929          $TRAN   'V785',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=785" - Add only
0929                  INCLUDE_CLASS,,,-       ;   11/785 systems to the display.
0929                  CLS$M_785_SYS           ;
0929
0929          $TRAN   'V750',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=750" - Add only
0929                  INCLUDE_CLASS,,,-       ;   11/750 systems to the display.
0929                  CLS$M_750_SYS           ;
0929
0929          $TRAN   'HS50',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=HS50" - Add only
0929                  INCLUDE_CLASS,,,-       ;   HSC50 systems to the display.
0929                  CLS$M_HSC50_SYS         ;
0929
0929          $TRAN   'HS60',TPA$_EXIT,-      ; "ADD SYSTEMS/TYPE=HS60" - Add only
0929                  INCLUDE_CLASS,,,-       ;   HSC60 systems to the display.
0929                  CLS$M_HSC60_SYS         ;
0929
0929          $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Any other type is an error.
0929
0929
```

```
0929 ;
0929 ; Process ADD SYSTEMS/NODE=nodename
0929 ;         ADD SYSTEMS/NODE=(nodename,...)
0929
0929         $STATE  ADD_SYS_NODE
0929         $TRAN   '='
0929         $STATE
0929         $TRAN   '(',ADD_NODE_LOOP          ; Go parse a list of nodenames
0929         $TRAN   TPA$_LAMBDA                ; Go parse a single nodename
0929         $STATE
0929         $TRAN   !ADD_SYS_NODE_STRING,-     ;
0929                 NEXT_ADD_SYS_QUAL          ;
0929
0929         $STATE  ADD_NODE_LOOP              ; Parse a list of nodenames till a ")"
0929                                            ;  is found.
0929         $TRAN   !ADD_SYS_NODE_STRING       ; Go parse a nodename.
0929         $STATE
0929         $TRAN   <','>,ADD_NODE_LOOP        ; Loop if "," found.
0929         $TRAN   ')',NEXT_ADD_SYS_QUAL      ; Until a ")" is found.
0929
0929         $STATE  ADD_SYS_NODE_STRING        ; Parse a single nodename
0929         $TRAN   TPA$_STRING,,,,TKN_DSCR    ; Save address of string.
0929         $STATE
0929         $TRAN   TPA$_LAMBDA,TPA$_EXIT,-    ; Call ADD_NODE.
0929                 ADD_NODE
0929
0929
```

```
0929       ;
0929       ; ADD CIRCUITS
0929       ;
0929               $STATE  ADD_CIR
0929
0929               $TRAN   '/',ADD_CIR_QUAL         ; Go to ADD_CIR_QUAL state if qualifier
0929                                                ;  is present.
0929
0929               $TRAN   TPA$_LAMBDA,-            ; Otherwise add all types of circuits
0929                       ADD_NEXT_CLASS,-         ;  to the display.
0929                       INCLUDE_CLASS,,,-        ;
0929                       CLS$M_ALL_CIR            ;
0929
0929       ;
0929       ; Process ADD CIRCUITS qualifiers.
0929       ;
0929               $STATE  ADD_CIR_QUAL
0929
0929               $TRAN   'ALL',-                  ; ADD CIRCUITS/ALL
0929                       NEXT_ADD_CIR_QUAL,-      ;
0929                       ADD_ALL_FIELDS,,,-       ;
0929                       CLS$M_ALL_CIR
0929               $TRAN   'TYPE',ADD_CIR_TYPE      ; ADD CIRCUITS/TYPE
0929               $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929
0929       ;
0929       ; Loop for multiple qualifiers on the CIRCUITS class.
0929       ;
0929               $STATE NEXT_ADD_CIR_QUAL
0929               $TRAN   '/',ADD_CIR_QUAL
0929               $TRAN   TPA$_LAMBDA,ADD_NEXT_CLASS
0929
0929               $STATE  ADD_CIR_TYPE
0929               $TRAN   '='
0929               $STATE
0929               $TRAN   '(',ADD_CIR_TYPE_LOOP    ; Go parse a list of circuit types.
0929               $TRAN   TPA$_LAMBDA,             ; Go parse a single circuit type.
0929               $STATE
0929               $TRAN   !ADD_CIR_TYPE_STRING,-   ;
0929                       NEXT_ADD_CIR_QUAL        ;
0929
0929               $STATE  ADD_CIR_TYPE_LOOP        ; Parse a list of circuit types till a
0929                                                ;  ")" is found.
0929               $TRAN   !ADD_CIR_TYPE_STRING     ;
0929               $STATE
0929               $TRAN   <','>,ADD_CIR_TYPE_LOOP  ; Go parse a single circuit type.
0929               $TRAN   ')',NEXT_ADD_CIR_QUAL    ; Exit loop if ")" found.
0929
0929
0929               $STATE  ADD_CIR_TYPE_STRING      ; Parse a single circuit type.
0929
0929               $TRAN   'ALL',TPA$_EXIT,-        ; "ADD CIRCUITS/TYPE=ALL" - Add all
0929                       INCLUDE_CLASS,,,-        ; types of circuits to the display.
0929                       CLS$M_ALL_CIR
0929
0929               $TRAN   'OPEN',TPA$_EXIT,-       ; "ADD CIRCUITS/OPEN" - Add circuits in
0929                       INCLUDE_CLASS,,,-        ;  the open state to the display.
```

```
0929                    CLS$M_CIR!CLS$M_OPEN_CIR;
0929
0929        $TRAN       'NOOPEN',TPA$_EXIT,-      ; "ADD CIRCUITS/NOOPEN" - Add circuits
0929                    INCLUDE_CLASS,,-          ;  in non-open states to the display.
0929                    CLS$M_CIR!CLS$M_NOOPEN_CIR
0929
0929        $TRAN       TPA$_LAMBDA,TPA$_FAIL     ; Any other type is an error.
0929
0929
```

```
0929  ;
0929  ; ADD CONNECTIONS
0929  ;
0929          $STATE  ADD_CON
0929
0929          $TRAN   '/',ADD_CON_QUAL          ; Go to ADD_CON_QUAL state if qualifier
0929                                            ;  is present.
0929
0929          $TRAN   TPA$_LAMBDA,-             ; Otherwise add all types of connections
0929                  ADD_NEXT_CLASS,-          ;  to the display.
0929                  INCLUDE_CLASS,,,-         ;
0929                  CLS$M_ALL_CON             ;
0929
0929  ;
0929  ; Process ADD CONNECTIONS qualifiers.
0929  ;
0929          $STATE  ADD_CON_QUAL
0929
0929          $TRAN   'ALL',-                   ; ADD CONNECTIONS/ALL
0929                  NEXT_ADD_CON_QUAL,-       ;
0929                  ADD_ALL_FIELDS,,,-        ;
0929                  CLS$M_ALL_CON
0929          $TRAN   'TYPE',ADD_CON_TYPE       ; ADD CONNECTIONS/TYPE
0929          $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929
0929  ;
0929  ; Loop for multiple qualifiers on the CONNECTIONS class.
0929  ;
0929          $STATE NEXT_ADD_CON_QUAL
0929          $TRAN   '/',ADD_CON_QUAL
0929          $TRAN   TPA$_LAMBDA,ADD_NEXT_CLASS
0929
0929          $STATE  ADD_CON_TYPE
0929          $TRAN   '='
0929          $STATE
0929          $TRAN   '(',ADD_CON_TYPE_LOOP     ; Go parse a list of connection types.
0929          $TRAN   TPA$_LAMBDA,              ; Go parse a single connection type.
0929          $STATE
0929          $TRAN   !ADD_CON_TYPE_STRING,-    ;
0929                  NEXT_ADD_CON_QUAL         ;
0929
0929
0929          $STATE  ADD_CON_TYPE_LOOP         ; Parse a list of connection types till
0929                                            ;  a ")" is found.
0929          $TRAN   !ADD_CON_TYPE_STRING      ;
0929          $STATE
0929          $TRAN   <','>,ADD_CON_TYPE_LOOP   ; Go parse a single connection type.
0929          $TRAN   ')',NEXT_ADD_CON_QUAL     ; Exit loop if ")" found.
0929
0929
0929          $STATE  ADD_CON_TYPE_STRING       ; Parse a single connection type.
0929
0929          $TRAN   'ALL',TPA$_EXIT,-         ; "ADD CONNECTION/TYPE=ALL" - Add all
0929                  INCLUDE_CLASS,,,-         ; types of connections to the display.
0929                  CLS$M_ALL_CON
0929
0929          $TRAN   'OPEN',TPA$_EXIT,-        ; "ADD CONNECTIONS/OPEN" - Add
```

```
                INCLUDE_CLASS,,,-              ;  connections in the open state to the
                CLS$M_CON!CLS$M_OPEN_CON;      display.

        $TRAN   'NOOPEN',TPA$_EXIT,-           ; "ADD CONNECTIONS/NOOPEN" - Add
                INCLUDE_CLASS,,,-              ;  connections in non-open states to
                CLS$M_CON!CLS$M_NOOPEN_CON  ;  the display.

        $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Any other type is an error.
```

```
0929        ;
0929        ; Translate the REMOVE "class" or "field-name" commands.
0929        ;
0929                $STATE  REMOVE
0929
0929                $TRAN   TPA$_EOS,TPA$_EXIT,-       ; Return if end of string.
0929                        REFRESH
0929
0929                $TRAN   'CLUSTER',REM_CLUSTER,-    ; "REMOVE CLUSTER"
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   'SYSTEMS',REM_SYS,-        ; "REMOVE SYSTEMS".
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   'MEMBERS',REM_MEMBERS,-    ; "REMOVE MEMBERS".
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   'CONNECTIONS',REM_CON,-    ; "REMOVE CONNECTIONS".
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   'CIRCUITS',REM_CIR,-       ; "REMOVE CIRCUITS".
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   'COUNTERS',-               ; "REMOVE COUNTERS".
0929                        REM_COUNTERS,-             ;
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   'CREDITS',REM_CREDITS,-    ; "REMOVE CREDITS".
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   'ERRORS',REM_ERRORS,-      ; "REMOVE ERRORS".
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   'LOCAL_PORTS',-            ; "REMOVE LOCAL_PORTS".
0929                        REM_LOCAL_PORTS,-          ;
0929                        ,,TKN_DSCR                 ;
0929
0929                $TRAN   TPA$_SYMBOL,,-             ; "REMOVE field-name".
0929                        REMOVE_FIELD,-             ;
0929                        ,,TKN_DSCR                 ;
0929
0929
0929                $STATE  REM_NEXT_CLASS
0929                $TRAN   <','>,REMOVE               ; Loop if another class was
0929                                                   ;  specified.
0929                $TRAN   '!',TPA$_EXIT,REFRESH      ; Allow comments...
0929                $TRAN   TPA$_EOS,TPA$_EXIT,-       ;
0929                        REFRESH                    ;  or end of string.
0929                $TRAN   TPA$_LAMBDA,TPA$_FAIL,-    ;
0929                        REFRESH                    ; Anything else is an error.
0929
0929
```

```
0929    ;
0929    ; REMOVE CLUSTER
0929    ;
0929
0929            $STATE  REM_CLUSTER
0929            $TRAN   TPA$_LAMBDA,-
0929                    REM_NEXT_CLASS,-
0929                    EXCLUDE_CLASS,,,-
0929                    CLS$M_CLUB
0929
0929    ;
0929    ; REMOVE MEMBERS
0929    ;
0929
0929            $STATE  REM_MEMBERS
0929            $TRAN   TPA$_LAMBDA,-
0929                    REM_NEXT_CLASS,-
0929                    EXCLUDE_CLASS,,,-
0929                    CLS$M_CSB
0929
0929    ;
0929    ; REMOVE COUNTERS
0929    ;
0929            $STATE  REM_COUNTERS
0929            $TRAN   TPA$_LAMBDA,-
0929                    REM_NEXT_CLASS,-
0929                    EXCLUDE_CLASS,,,-
0929                    CLS$M_CNT
0929
0929    ;
0929    ; REMOVE CREDITS
0929    ;
0929            $STATE  REM_CREDITS
0929            $TRAN   TPA$_LAMBDA,-
0929                    REM_NEXT_CLASS,-
0929                    EXCLUDE_CLASS,,,-
0929                    CLS$M_CRD
0929
0929    ;
0929    ; REMOVE LOCAL_PORTS
0929    ;
0929
0929            $STATE  REM_LOCAL_PORTS
0929            $TRAN   TPA$_LAMBDA,-
0929                    REM_NEXT_CLASS,-
0929                    EXCLUDE_CLASS,,,-
0929                    CLS$M_LPRT
0929
0929    ;
0929    ; REMOVE ERRORS
0929    ;
0929
0929            $STATE  REM_ERRORS
0929            $TRAN   TPA$_LAMBDA,-
0929                    REM_NEXT_CLASS,-
0929                    EXCLUDE_CLASS,,,-
0929                    CLS$M_ERR
```

```
0929 ;
0929 ; REMOVE SYSTEMS
0929 ;
0929         $STATE  REM_SYS
0929
0929         $TRAN   '/',REM_SYS_QUAL          ; Go to REM_SYS_QUAL state if
0929                                           ;  qualifier is present.
0929         $TRAN   TPA$_LAMBDA,-             ; Otherwise just REMOVE SYSTEMS.
0929                 REM_NEXT_CLASS,-          ;
0929                 EXCLUDE_CLASS,,,-         ;
0929                 CLS$M_SYS                 ;
0929
0929 ;
0929 ; Process REMOVE SYSTEMS qualifiers.
0929 ;
0929         $STATE  REM_SYS_QUAL
0929
0929         $TRAN   'ID',REM_SYS_ID           ; REMOVE SYSTEM/ID
0929         $TRAN   'TYPE',REM_SYS_TYPE       ; REMOVE SYSTEM/TYPE
0929         $TRAN   'NODE',REM_SYS_NODE       ; REMOVE SYSTEM/NODE
0929         $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929
0929 ;
0929 ; Loop for multiple qualifiers on the SYSTEMS class.
0929 ;
0929         $STATE NEXT_REM_SYS_QUAL
0929         $TRAN   '/',REM_SYS_QUAL
0929         $TRAN   TPA$_LAMBDA,REM_NEXT_CLASS
0929
0929
```

```
0929        ;
0929        ; Process REMOVE SYSTEMS/ID=system_id
0929        ;         REMOVE SYSTEMS/ID=(system_id,...)
0929
0929                $STATE  REM_SYS_ID
0929                $TRAN   '='
0929                $STATE
0929                $TRAN   '(',REM_ID_LOOP              ; Go parse a list of ID's
0929                $TRAN   TPA$_LAMBDA,                 ; Go parse a single ID
0929                $STATE
0929                $TRAN   !REM_SYS_ID_STRING,-         ;
0929                        NEXT_REM_SYS_QUAL            ;
0929
0929                $STATE  REM_ID_LOOP                  ; Parse a list of ID numbers till a ")"
0929                                                     ;  is found.
0929                $TRAN   !REM_SYS_ID_STRING           ;
0929                $STATE
0929                $TRAN   <','>,REM_ID_LOOP            ; Go parse a single ID
0929                $TRAN   ')',NEXT_REM_SYS_QUAL        ; Exit loop if ")" found.
0929
0929                $STATE  REM_SYS_ID_STRING            ; Parse a single ID
0929
0929                $TRAN   TPA$_STRING,,,,TKN_DSCR      ; Save address of string.
0929                $STATE
0929                $TRAN   TPA$_LAMBDA,TPA$_EXIT,-      ; Call ADD_ID to add system to
0929                        ADD_ID                       ;  disabled list.
0929
0929
```

```
0929    ;
0929    ; Process REMOVE SYSTEMS/TYPE=hw_type
0929    ;         REMOVE SYSTEMS/TYPE=(hw_type,...)
0929
0929            $STATE  REM_SYS_TYPE
0929            $TRAN   '='
0929            $STATE
0929            $TRAN   '(',REM_SYS_TYPE_LOOP   ; Go parse a list of system types.
0929            $TRAN   TPA$_LAMBDA,            ; Go parse a single system type.
0929            $STATE
0929            $TRAN   !REM_SYS_TYPE_STRING,-  ;
0929                    NEXT_REM_SYS_QUAL       ;
0929
0929            $STATE  REM_SYS_TYPE_LOOP       ; Parse a list of system types till a
0929                                            ;   ")" is found.
0929            $TRAN   !REM_SYS_TYPE_STRING    ;
0929            $STATE
0929            $TRAN   <','>,REM_SYS_TYPE_LOOP ; Go parse a single system type.
0929            $TRAN   ')',NEXT_REM_SYS_QUAL   ; Exit loop if ")" found.
0929
0929
0929            $STATE  REM_SYS_TYPE_STRING     ; Parse a single system type
0929
0929            $TRAN   'ALL',TPA$_EXIT,-       ; "REMOVE SYSTEMS/TYPE=ALL" - Remove all
0929                    EXCLUDE_CLASS,,,-       ; types of systems from the display.
0929                    CLS$M_ALL_SYS
0929
0929            $TRAN   'V780',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=780" - Remove
0929                    EXCLUDE_CLASS,,,-       ;  only 11/780 systems from the display.
0929                    CLS$M_780_SYS           ;
0929
0929            $TRAN   'V785',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=785" - Remove
0929                    EXCLUDE_CLASS,,,-       ;  only 11/785 systems from the display.
0929                    CLS$M_785_SYS           ;
0929
0929            $TRAN   'V750',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=750" - Remove
0929                    EXCLUDE_CLASS,,,-       ;  only 11/750 systems from the display.
0929                    CLS$M_750_SYS           ;
0929
0929            $TRAN   'HS50',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=HS50" - Remove
0929                    EXCLUDE_CLASS,,,-       ;  only HSC50 systems from the display.
0929                    CLS$M_HSC50_SYS         ;
0929
0929            $TRAN   'HS60',TPA$_EXIT,-      ; "REMOVE SYSTEMS/TYPE=HS60" - Remove
0929                    EXCLUDE_CLASS,,,-       ;  only HSC60 systems from the display.
0929                    CLS$M_HSC60_SYS         ;
0929
0929            $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Any other type is an error.
0929
0929
```

```
0929  ;
0929  ; Process REMOVE SYSTEMS/NODE=nodename
0929  ;         REMOVE SYSTEMS/NODE=(nodename,...)
0929
0929          $STATE  REM_SYS_NODE
0929          $TRAN   '='
0929          $STATE
0929          $TRAN   '(',REM_NODE_LOOP       ; Go parse a list of nodenames
0929          $TRAN   TPA$_LAMBDA             ; Go parse a single nodename
0929          $STATE
0929          $TRAN   !REM_SYS_NODE_STRING,-  ;
0929                  NEXT_REM_SYS_QUAL       ;
0929
0929          $STATE  REM_NODE_LOOP           ; Parse a list of nodenames till a ")"
0929                                          ;  is found.
0929          $TRAN   !REM_SYS_NODE_STRING    ; Go parse a nodename.
0929          $STATE
0929          $TRAN   <','>,REM_NODE_LOOP     ; Loop if "," found.
0929          $TRAN   ')',NEXT_REM_SYS_QUAL   ; Until a ")" is found.
0929
0929          $STATE  REM_SYS_NODE_STRING     ; Parse a single nodename
0929          $TRAN   TPA$_STRING,,,,TKN_DSCR ; Save address of string.
0929          $STATE
0929          $TRAN   TPA$_LAMBDA,TPA$_EXIT,- ; Call REM_NODE.
0929                  REM_NODE
0929
```

```
0929 ;
0929 ; REMOVE CIRCUITS
0929 ;
0929         $STATE  REM_CIR
0929
0929         $TRAN   '/',REM_CIR_QUAL            ; Go to REM_CIR_QUAL state if qualifier
0929                                             ;  is present.
0929
0929         $TRAN   TPA$_LAMBDA,-               ; Otherwise remove all types of circuits
0929                 REM_NEXT_CLASS,-            ;  to the display.
0929                 EXCLUDE_CLASS,,,-           ;
0929                 CLS$M_ALL_CIR               ;
0929
0929 ;
0929 ; Process REMOVE CIRCUITS qualifiers.
0929 ;
0929         $STATE  REM_CIR_QUAL
0929
0929         $TRAN   'TYPE',REM_CIR_TYPE         ; REMOVE CIRCUITS/TYPE
0929         $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929
0929 ;
0929 ; Loop for multiple qualifiers on the CIRCUITS class.
0929 ;
0929         $STATE NEXT_REM_CIR_QUAL
0929         $TRAN   '/',REM_CIR_QUAL
0929         $TRAN   TPA$_LAMBDA,REM_NEXT_CLASS
0929
0929         $STATE  REM_CIR_TYPE
0929         $TRAN   '='
0929         $STATE
0929         $TRAN   '(',REM_CIR_TYPE_LOOP       ; Go parse a list of circuit types.
0929         $TRAN   TPA$_LAMBDA,                ; Go parse a single circuit type.
0929         $STATE
0929         $TRAN   !REM_CIR_TYPE_STRING,-      ;
0929                 NEXT_REM_CIR_QUAL           ;
0929
0929
0929         $STATE  REM_CIR_TYPE_LOOP           ; Parse a list of circuit types till a
0929                                             ;  ")" is found.
0929         $TRAN   !REM_CIR_TYPE_STRING        ;
0929         $STATE
0929         $TRAN   <','>,REM_CIR_TYPE_LOOP     ; Go parse a single circuit type.
0929         $TRAN   ')',NEXT_REM_CIR_QUAL       ; Exit loop if ")" found.
0929
0929
0929         $STATE  REM_CIR_TYPE_STRING         ; Parse a single circuit type.
0929
0929         $TRAN   'ALL',TPA$_EXIT,-           ; "REMOVE CIRCUITS/TYPE=ALL" - Remove
0929                 EXCLUDE_CLASS,,,-           ;  all types of circuits from the
0929                 CLS$M_ALL_CIR               ;  display.
0929
0929         $TRAN   'OPEN',TPA$_EXIT,-          ; "REMOVE CIRCUITS/OPEN" - Remove
0929                 EXCLUDE_CLASS,,,-           ;  circuits in the open state from to
0929                 CLS$M_OPEN_CIR              ;  the display.
0929
0929         $TRAN   'NOOPEN',TPA$_EXIT,-        ; "REMOVE CIRCUITS/NOOPEN" - Remove
```

```
0929 11269                  EXCLUDE_CLASS,,,-          ;  circuits in non-open states from the
0929 11270                  CLS$M_NOOPEN_CIR           ; display.
0929 11271
0929 11272          $TRAN   TPA$_LAMBDA,TPA$_FAIL      ; Any other type is an error.
0929 11273
```

```
0929       ;
0929       ; REMOVE CONNECTIONS
0929       ;
0929               $STATE  REM_CON
0929
0929               $TRAN   '/',REM_CON_QUAL          ; Go to REM_CON_QUAL state if qualifier
0929                                                 ;  is present.
0929
0929               $TRAN   TPA$_LAMBDA,-             ; Otherwise remove all types of
0929                       REM_NEXT_CLASS,-          ;  connections from the display.
0929                       EXCLUDE_CLASS,,,-         ;
0929                       CLS$M_ALL_CON             ;
0929
0929       ;
0929       ; Process REMOVE CONNECTIONS qualifiers
0929       ;
0929               $STATE  REM_CON_QUAL
0929
0929               $TRAN   'TYPE',REM_CON_TYPE       ; REMOVE CONNECTIONS/TYPE
0929               $TRAN   TPA$_LAMBDA,TPA$_FAIL
0929
0929       ;
0929       ; Loop for multiple qualifiers on the CONNECTIONS class.
0929       ;
0929               $STATE NEXT_REM_CON_QUAL
0929               $TRAN   '/',REM_CON_QUAL
0929               $TRAN   TPA$_LAMBDA,REM_NEXT_CLASS
0929
0929               $STATE  REM_CON_TYPE
0929               $TRAN   '='
0929               $STATE
0929               $TRAN   '(',REM_CON_TYPE_LOOP     ; Go parse a list of connection types.
0929               $TRAN   TPA$_LAMBDA,              ; Go parse a single connection type.
0929               $STATE
0929               $TRAN   !REM_CON_TYPE_STRING,-    ;
0929                       NEXT_REM_CON_QUAL         ;
0929
0929
0929               $STATE  REM_CON_TYPE_LOOP         ; Parse a list of connection types till
0929                                                 ;  a ")" is found.
0929               $TRAN   !REM_CON_TYPE_STRING      ;
0929               $STATE
0929               $TRAN   <','>,REM_CON_TYPE_LOOP   ; Go parse a single connection type.
0929               $TRAN   ')',NEXT_REM_CON_QUAL     ; Exit loop if ")" found.
0929
0929
0929               $STATE  REM_CON_TYPE_STRING       ; Parse a single connection type.
0929
0929               $TRAN   'ALL',TPA$_EXIT,-         ; "REM CONNECTION/TYPE=ALL" - Remove all
0929                       EXCLUDE_CLASS,,,-         ;  types of connections from the
0929                       CLS$M_ALL_CON             ;  display.
0929
0929               $TRAN   'OPEN',TPA$_EXIT,-        ; "REMOVE CONNECTIONS/OPEN" - Remove
0929                       EXCLUDE_CLASS,,,-         ;  connections in the open state from
0929                       CLS$M_OPEN_CON            ;  the display.
0929
0929               $TRAN   'NOOPEN',TPA$_EXIT,-      ; "REMOVE CONNECTIONS/NOOPEN" - Remove
```

```
0929                       EXCLUDE_CLASS,,,-         ;  connections in non-open states from
0929                       CLS$M_NOOPEN_CON          ;  the display.
0929
0929               $TRAN   TPA$_LAMBDA,TPA$_FAIL     ; Any other type is an error.
0929
0929
```

```
;
; Set command.
;

        $STATE  SET
        $TRAN   TPA$_EOS,TPA$_EXIT,-    ; Return if end of string.
                REFRESH

        $TRAN   'INTERVAL',-            ; SET INTERVAL
                PARSE_INTERVAL          ;
        $TRAN   'SCREEN',PARSE_SCREEN   ; SET SCREEN
        $TRAN   TPA$_LAMBDA,SET_FIELD   ; Must be SET FIELD


        $STATE  SET_FIELD
        $TRAN   TPA$_SYMBOL,-           ; Check for valid field name.
                SET_FIELD_QUAL,-        ;
                LOOKUP_FIELD            ;

        $STATE NEXT_SET_FIELD_QUAL
        $TRAN   '/',SET_FIELD_QUAL      ; Loop on qualifiers to the fieldname.
        $TRAN   TPA$_LAMBDA,SET_NEXT    ; Or else go on to next SET entity.

        $STATE  SET_FIELD_QUAL
        $TRAN   '/'                     ; Expect a qualifier.
        $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Else error.
        $STATE
        $TRAN   'WIDTH',SET_FIELD_WIDTH ; SET field-name/WIDTH.
        $TRAN   'FORMAT',-              ; SET field-name/FORMAT.
                SET_FIELD_FORMAT        ;
        $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; Any other qualifier is an error.

        $STATE  SET_FIELD_WIDTH         ; SET field/WIDTH
        $TRAN   !PARSE_VALUE,-          ; Get the value and set it.
                NEXT_SET_FIELD_QUAL,-   ;
                SET_WIDTH               ;

        $STATE  SET_FIELD_FORMAT        ; Get the desired field format.
        $TRAN   '='                     ; SET field/FORMAT=
        $STATE
        $TRAN   'DECIMAL',-             ; SET field/FORMAT=DECIMAL
                NEXT_SET_FIELD_QUAL,-   ;
                SET_FORMAT,,,DEC
        $TRAN   'HEXADECIMAL',-         ; SET field/FORMAT=HEXADECIMAL
                NEXT_SET_FIELD_QUAL,-   ;
                SET_FORMAT,,,HEX        ;

        $STATE  SET_NEXT
        $TRAN   <','>,SET               ; Loop on set command if comma follows
                                        ;  the entity.
        $TRAN   TPA$_LAMBDA,TPA$_EXIT,- ; If not comma then redraw the header
                REFRESH                 ;  and return.

        $STATE  PARSE_INTERVAL
        $TRAN   !PARSE_VALUE,TPA$_EXIT,-; Get the value and set it.
                SET_INTERVAL            ;
```

```
0929          $STATE  PARSE_SCREEN
0929          $TRAN   !PARSE_VALUE,TPA$_EXIT,-; Get the value and set it.
0929                  SET_SCREEN              ;
0929
0929          $STATE  PARSE_VALUE
0929          $TRAN   '='                     ; Must begin with =
0929          $STATE                          ;
0929          $TRAN   <'-'>,TPA$_FAIL         ; Negative numbers are a no-no.
0929          $TRAN   TPA$_DECIMAL,TPA$_EXIT,-; Store a decimal number at VALUE.
0929                  ,,VALUE                 ;
0929          $TRAN   TPA$_LAMBDA,TPA$_FAIL   ; All else fails.
0929
0929
0929   ;
0929   ; Help command
0929   ;
0929
0929          $STATE  HELP
0929
0929          $TRAN   TPA$_LAMBDA,TPA$_EXIT,GET_HELP
0929
0929          $END_STATE
0929
0929
```

```
                        0929        .SBTTL  Tparse action routines.
                        0929        .SBTTL          -           SHOW_CLUSTER
                        0929 ;++
                        0929 ;
                        0929 ; FUNCTIONAL DESCRIPTION:
                        0929 ;
                        0929 ;      This routine sets the report to clusters.  It stores the parameters
                        0929 ;      necessary for the FIND_ENTRY routine to locate a field descriptor for
                        0929 ;      the cluster report.  It sets the bit in CURRENT_REPORT that indicates
                        0929 ;      that the cluster report is being displayed.  It also saves a copy of
                        0929 ;      the scroll flags for the local report and restores the scroll flags
                        0929 ;      for the cluster report.
                        0929 ;
                        0929 ;
                        0929 ; CALLING SEQUENCE:
                        0929 ;
                        0929 ;      CALLS   #0,SHOW_CLUSTER
                        0929 ;
                        0929 ; CALLED BY:
                        0929 ;
                        0929 ;      SWITCH_DISPLAY
                        0929 ;      RESET
                        0929 ;
                        0929 ;
                        0929 ; IMPLICIT INPUTS:
                        0929 ;
                        0929 ;      CLSTR_SCROLL_FLAGS
                        0929 ;                      contains a mask of the scroll flags set from the last
                        0929 ;                      time the cluster report was being displayed.
                        0929 ;
                        0929 ; IMPLICIT OUTPUTS:
                        0929 ;
                        0929 ;      ADRS_TABLE      contains the address of the cluster address table.
                        0929 ;      BITMAP_ADRS     contains the address of the cluster bitmap.
                        0929 ;      KEYTAB_ADRS     contains the address of the cluster keyword table.
                        0929 ;      MAP_LEN         contains the the length of the cluster bitmap.
                        0929 ;      CURRENT_REPORT  Set to clusters by setting RPT$M_REPORT.
                        0929 ;      SCROLL_FLAGS    restored to from the last time the cluster report was
                        0929 ;                      displayed.
                        0929 ;
                        0929 ; RETURN STATUS:
                        0929 ;
                        0929 ;      SS$_NORMAL or failed status returned in R0.
                        0929 ;
                        0929 ;--
                        0929
               0000     0929 .ENTRY  SHOW_CLUSTER,^M<>
                        092B
          1AA2'   30    092B        BSBW    CLEAR_SCREEN            ; Clear the screen.
          6D 50   E9    092E        BLBC    R0,99$                  ; Branch on error.
      0000055B'EF 7F    0931        PUSHAQ  SCREEN_DSCR             ; Set up a buffer for the screen.
  00000000'GF 01  FB    0937        CALLS   #1,G^LIB$SET_BUFFER     ;
          5D 50   E9    093E        BLBC    R0,99$                  ; Branch on error.
             00   E2    0941        BBSS    #RPT$V_CLSTR,-          ; Branch if we're already displaying
     50 F6BA CF         0943                CURRENT_REPORT,10$      ;  the cluster report.  Otherwise set
                        0947                                        ;  the current report to clusters.
      0000047A'EF 90    0947        MOVB    SCROLL_FLAGS,-          ; Store the scroll flags for the local
```

```
                    0000047B'EF        094D 11478               LOCAL_SCROLL_FLAGS         ;  report.
                    0000047C'EF    90  0952 11479        MOVB   CLSTR_SCROLL_FLAGS,-       ; Get the cluster report scroll flags.
                    0000047A'EF        0958 11480               SCROLL_FLAGS               ;
       00002200'EF     F6A0 CF     D0  095D 11481        MOVL   CLASS_SELECT,LOCAL_SAVE    ; Save local report class selections.
         F692 CF    00002204'EF    D0  0966 11482        MOVL   CLUST_SAVE,CLASS_SELECT    ; Get cluster report class selections.
                    00000000'EF    DE  096F 11483        MOVAL  CLUSTER_KEYTAB,-           ; Store address of cluster keyword
                    00002274'EF        0975 11484               KEYTAB_ADRS                ;  table.
   00002264'EF      00000000'EF    DE  097A 11485        MOVAL  CLUSTER_TABLE,ADRS_TABLE;  Store address of cluster address
                                       0985 11486                                          ;  table.
                    00000000'EF    DE  0985 11487        MOVAL  CLUSTER_BITMAP,-           ; Store address of cluster bitmap.
                    00002268'EF        098B 11488               BITMAP_ADRS                ;
            0000226C'EF     02     D0  0990 11489        MOVL   #CLUSTER_MAP_LEN,MAP_LEN;  Store length of cluster bitmap.
            00002348'EF     00     FB  0997 11490 10$:   CALLS  #0,REFRESH                 ; Redraw the header.
                                   04  099E 11491 99$:   RET
                                       099F 11492
```

```
                                  099F        .SBTTL          -       SHOW_LOCAL
                                  099F  ;++
                                  099F  ;
                                  099F  ; FUNCTIONAL DESCRIPTION:
                                  099F  ;
                                  099F  ;       This routine sets the report to local ports.  It stores the parameters
                                  099F  ;       necessary for the FIND_ENTRY routine to locate a field descriptor for
                                  099F  ;       the local report.  It sets the bit in CURRENT_REPORT that indicates
                                  099F  ;       that the local report is being displayed.  It also saves a copy of
                                  099F  ;       the scroll flags for the cluster report and restores the scroll flags
                                  099F  ;       for the local report.
                                  099F  ;
                                  099F  ; CALLING SEQUENCE:
                                  099F  ;
                                  099F  ;       CALLS   #0,SHOW_LOCAL
                                  099F  ;
                                  099F  ; CALLED BY:
                                  099F  ;
                                  099F  ;       SWITCH_DISPLAY
                                  099F  ;
                                  099F  ; IMPLICIT INPUTS:
                                  099F  ;
                                  099F  ;       LOCAL_SCROLL_FLAGS
                                  099F  ;                       contains a mask of the scroll flags set from the last
                                  099F  ;                       time the cluster report was being displayed.
                                  099F  ;
                                  099F  ; IMPLICIT OUTPUTS:
                                  099F  ;
                                  099F  ;       ADRS_TABLE      contains the address of the local status address table.
                                  099F  ;       BITMAP_ADRS     contains the address of the local status bitmap.
                                  099F  ;       KEYTAB_ADRS     contains the address of the local status keyword table.
                                  099F  ;       MAP_LEN         contains the the length of the local status bitmap.
                                  099F  ;       CURRENT_REPORT  Set to local report by clearing RPT$M_REPORT.
                                  099F  ;       SCROLL_FLAGS    restored to from the last time the cluster report was
                                  099F  ;                       displayed.
                                  099F  ;
                                  099F  ;
                                  099F  ; RETURN STATUS:
                                  099F  ;
                                  099F  ;       SS$_NORMAL or failed status returned in R0.
                                  099F  ;
                                  099F  ;--
                                  099F
                          0000    099F  .ENTRY  SHOW_LOCAL,^M<>
                                  09A1
                   1A2C'   30     09A1          BSBW    CLEAR_SCREEN            ; Clear the screen.
                   6D 50   E9     09A4          BLBC    R0,99$                  ; Branch on error.
            0000055B'EF    7F     09A7          PUSHAQ  SCREEN_DSCR             ; Set up a buffer for the screen.
       00000000'GF  01     FB     09AD          CALLS   #1,G^LIB$SET_BUFFER     ;
                   5D 50   E9     09B4          BLBC    R0,99$                  ; Branch on error.
                      00   E5     09B7          BBCC    #RPT$V_CLSTR,-          ; Branch if we are already displaying
               50 F644 CF         09B9                  CURRENT_REPORT,10$      ;  local report.  Otherwise, set the
                                  09BD                                          ;  current report to local.
            0000047A'EF    90     09BD          MOVB    SCROLL_FLAGS,-          ; Store the current scroll flags for
            0000047C'EF           09C3                  CLSTR_SCROLL_FLAGS      ;  the cluster report.
            0000047B'EF    90     09C8          MOVB    LOCAL_SCROLL_FLAGS,-    ; Get the local report scroll flags.
            0000047A'EF           09CE                  SCROLL_FLAGS            ;
 00002204'EF    F62A CF    D0     09D3          MOVL    CLASS_SELECT,CLUST_SAVE ; Save cluster report class selections.
```

```
     F61C'CF    00002200'EF  D0  09DC 11551         MOVL    LOCAL_SAVE,CLASS_SELECT ; Get local report class selections.
00002274'EF     00000000'EF  DE  09E5 11552         MOVAL   LOCAL_KEYTAB,KEYTAB_ADRS; Store address of local status keyword
                                 09F0 11553                                         ;  table.
00002264'EF     00000000'EF  DE  09F0 11554         MOVAL   LOCAL_TABLE,ADRS_TABLE  ; Store address of local table.
00002268'EF     00000000'EF  DE  09FB 11555         MOVAL   LOCAL_BITMAP,BITMAP_ADRS; Store address of local bitmap.
          0000226C'EF     00 D0  0A06 11556         MOVL    #LOCAL_MAP_LEN,MAP_LEN  ; Store length of local bitmap.
          00002348'EF     00 FB  0A0D 11557 10$:    CALLS   #0,REFRESH              ; Redraw the header.
                             04  0A14 11558 99$:    RET
                                 0A15 11559
                                 0A15 11560
```

```
                              0A15         .SBTTL          -          ADD_FIELD
                              0A15   ;++
                              0A15   ;
                              0A15   ; FUNCTIONAL DESCRIPTION:
                              0A15   ;
                              0A15   ;       This routine is used to add a field to the display by setting the bit
                              0A15   ;       corresponding to the field in the bitmap for the current report.  If
                              0A15   ;       the field is in a class which is currently not being displayed, then
                              0A15   ;       that class is enabled along with the desired field, but all other
                              0A15   ;       fields in that class are turned off.  If the class being enabled is
                              0A15   ;       the Cluster class, then update FIRST_DATA_LINE so that the scrolling
                              0A15   ;       region does not encompass the Cluster class display.
                              0A15   ;
                              0A15   ; CALLING SEQUENCE:
                              0A15   ;
                              0A15   ;       CALLS   #0,ADD_FIELD
                              0A15   ;
                              0A15   ; CALLED_BY:
                              0A15   ;
                              0A15   ;       READ_W_PROMPT
                              0A15   ;
                              0A15   ; INPUTS:
                              0A15   ;
                              0A15   ;       KEY_OFF = Address of longword in the bitmap that contains the bit
                              0A15   ;                 corresponding to the field selected.
                              0A15   ;       KEY_MSK = Mask of the bit within KEY_OFF corresponding to the field.
                              0A15   ;
                              0A15   ; OUTPUTS:
                              0A15   ;
                              0A15   ;       Bit corresponding to the field set in the bitmap.
                              0A15   ;
                              0A15   ; RETURN STATUS:
                              0A15   ;
                              0A15   ;       SS$_NORMAL or failed status returned in R0.
                              0A15   ;
                              0A15   ; REGISTER USAGE:
                              0A15   ;
                              0A15   ;       R2 -    Points to the address containing the bit corresponding to the
                              0A15   ;               field.
                              0A15   ;
                              0A15   ;--
                              0A15
                        01FC  0A15   .ENTRY  ADD_FIELD,^M<R2,R3,R4,R5,R6,R7,R8>
  00000B3A'EF     00      FB  0A17           CALLS   #0,LOOKUP_FIELD           ; Check for valid field name.
                              0A1E           JLBC    R0,99$                    ; Branch on error.
  52    00000543'EF       D0  0A24           MOVL    KEY_OFF,R2                ; Get address of longword containing
                              0A2B                                             ;  the bit.
  53    0000054B'EF       D0  0A2B           MOVL    KEY_DSCR,R3               ; Get the address of the descriptor for
                              0A32                                             ;  the field.
              0C A3       D3  0A32           BITL    FD$L_CLASS(R3),-          ; Is the class for this field already
            F5C9 CF           0A35                   CLASS_SELECT              ;  enabled?
                              0A38           JNEQ    70$                       ; Branch if so.
                 03       E1  0A3D           BBC     #CLS$V_CIR,-              ; Is this field in the circuits class?
           0B 0C A3           0A3F                   FD$L_CLASS(R3),10$        ;
       00000608 8F        C8  0A42           BISL    #CLS$M_ALL_CIR,-          ; If so, enable all types of circuits.
            F5B6 CF           0A48                   CLASS_SELECT              ;
                 22       11  0A4B           BRB     30$                       ; Jump to common code.
```

```
                 04   E1  0A4D 11619 10$:   BBC     #CLS$V_CON,-                  ; Is this field in the connections
           0B 0C A3       0A4F 11620                FD$L_CLASS(R3),20$            ;  class?
        00001810 8F   C8  0A52 11621                BISL    #CLS$M_ALL_CON,-      ; If so, enable all types of
             F5A6 CF      0A58 11622                CLASS_SELECT                  ;  connections.
                 12   11  0A5B 11623                BRB     30$                   ; Jump to common code.
                 00   E1  0A5D 11624 20$:   BBC     #CLS$V_CLUB,-                 ; Is this field in the cluster class?
           07 0C A3       0A5F 11625                FD$L_CLASS(R3),25$            ;
   00002254'EF   0E   D0  0A62 11626        MOVL    #14,FIRST_DATA_LINE           ; If so, set first line to be scrolled.
              0C A3   C8  0A69 11627 25$:   BISL    FD$L_CLASS(R3),-              ; Enable the class.
             F592 CF      0A6C 11628                CLASS_SELECT                  ;
58      00002268'EF   D0  0A6F 11629 30$:   MOVL    BITMAP_ADRS,R8                ; Get the address of field descriptor
                          0A76 11630                                              ;  bitmap for the current report.
        00002270'EF   D4  0A76 11631        CLRL    START_BIT                     ; Set up for FIND_ENTRY.
        0000046C'EF   D4  0A7C 11632        CLRL    H_SCROLL_LINE                 ; Make sure that fields scrolled off
                          0A82 11633                                              ;  the screen are also checked.
                 00   E1  0A82 11634        BBC     #RPT$V_CLSTR,-                ; Branch if we're not displaying the
          09 F579 CF      0A84 11635                CURRENT_REPORT,50$            ;  cluster report.
   0000226C'EF   02   D0  0A88 11636        MOVL    #CLUSTER_MAP_LEN,MAP_LEN;     ; Save length of cluster bitmap.
                 07   11  0A8F 11637        BRB     60$                           ; Jump to common code.
   0000226C'EF   00   D0  0A91 11638 50$:   MOVL    #LOCAL_MAP_LEN,MAP_LEN        ; Save length of local bitmap.
               18C8'  30  0A98 11639 60$:   BSBW    FIND_ENTRY                    ; Search for any field being displayed.
              1D 50   E9  0A9B 11640        BLBC    R0,70$                        ; Branch if none found.
              0C AA   D1  0A9E 11641        CMPL    FD$L_CLASS(R10),-             ; Is the class for the field we just
              0C A3       0AA1 11642                FD$L_CLASS(R3)                ;  found the same as the field we are
                          0AA3 11643                                              ;  adding to the display?
                 F3   12  0AA3 11644        BNEQ    60$                           ; If not go find another field.
54   00002270'EF   01   C3  0AA5 11645      SUBL3   #1,START_BIT,R4               ; Get the number of the bit in the
                          0AAD 11646                                              ;  bitmap that corresponds to the field
                          0AAD 11647                                              ;  we just found.
                 55   D4  0AAD 11648        CLRL    R5                            ; Set up for extended divide.
  57    56    54   08   7B  0AAF 11649      EDIV    #8,R4,R6,R7                   ; R6 gets the index into the address
                          0AB4 11650                                              ;  table.  R7 gets the bit position.
        DF 6846  57   E5  0AB4 11651        BBCC    R7,(R8)[R6],60$               ; Disable the field and loop.
                 DD   11  0AB9 11652        BRB     60$                           ; Loop for all field descriptors.
62      00000547'EF   C8  0ABB 11653 70$:   BISL    KEY_MSK,(R2)                  ; Set the bit, turn on the field.
              50   01   D0  0AC2 11654      MOVL    #SS$_NORMAL,R0
                      04  0AC5 11655 99$:   RET
                          0AC6 11656
                          0AC6 11657
```

```
                              0AC6          .SBTTL          -         REMOVE_FIELD
                              0AC6   ;++
                              0AC6   ;
                              0AC6   ; FUNCTIONAL DESCRIPTION:
                              0AC6   ;
                              0AC6   ;          This routine is used to remove a field from the display by clearing the
                              0AC6   ;          bit corresponding to the field in the bitmap for the current report.
                              0AC6   ;          If the field being cleared is the last field being displayed in a
                              0AC6   ;          particular class, then the class is turned off instead.  If the class
                              0AC6   ;          being turned off is the Cluster class, then update FIRST_DATA_LINE so
                              0AC6   ;          that the scrolling region uses the full display.
                              0AC6   ;
                              0AC6   ; CALLING SEQUENCE:
                              0AC6   ;
                              0AC6   ;          CALLS   #0,REMOVE_FIELD
                              0AC6   ;
                              0AC6   ; CALLED_BY:
                              0AC6   ;
                              0AC6   ;          READ_W_PROMPT
                              0AC6   ;
                              0AC6   ; INPUTS:
                              0AC6   ;
                              0AC6   ;          KEY_OFF = Address of longword in the bitmap that contains the bit
                              0AC6   ;                    corresponding to the field selected.
                              0AC6   ;          KEY_MSK = Mask of the bit within KEY_OFF corresponding to the field.
                              0AC6   ;
                              0AC6   ; OUTPUTS:
                              0AC6   ;
                              0AC6   ;          Bit corresponding to the field cleared in the bitmap, or bit
                              0AC6   ;          corresponding to the class cleared at CLASS_SELECT.
                              0AC6   ;
                              0AC6   ; RETURN STATUS:
                              0AC6   ;
                              0AC6   ;          SS$_NORMAL or failed status returned in R0.
                              0AC6   ;
                              0AC6   ; REGISTER USAGE:
                              0AC6   ;
                              0AC6   ;          R2 -    Points to the address containing the bit corresponding to the
                              0AC6   ;                  field.
                              0AC6   ;
                              0AC6   ;--
                        010C  0AC6   .ENTRY   REMOVE_FIELD,^M<R2,R3,R8>
 00000B3A'EF    00        FB  0AC8            CALLS   #0,LOOKUP_FIELD            ; Check for valid field name.
           67   50        E9  0ACF            BLBC    R0,99$                     ; Branch on error.
 52    00000543'EF        D0  0AD2            MOVL    KEY_OFF,R2                 ; Get address of longword containing
                              0AD9                                               ;  the bit.
 53    0000054B'EF        D0  0AD9            MOVL    KEY_DSCR,R3                ; Get the address of the descriptor
                              0AE0                                               ;  for the field.
 58    00002268'EF        D0  0AE0            MOVL    BITMAP_ADRS,R8             ; Get the address of the field
                              0AE7                                               ;  descriptor bitmap for the current
                              0AE7                                               ;  report.
       00002270'EF        D4  0AE7            CLRL    START_BIT                  ; Set up for FIND_ENTRY.
       0000046C'EF        D4  0AED            CLRL    H_SCROLL_LINE              ; Make sure that fields scrolled off
                              0AF3                                               ;  the screen are also checked.
                  00      E1  0AF3            BBC     #RPT$V_CLSTR,-             ; Branch if we're not displaying the
            09 F508       CF  0AF5                    CURRENT_REPORT,20$         ;  cluster report.
 0000226C'EF    02        D0  0AF9            MOVL    #CLUSTER_MAP_LEN,MAP_LEN;  Save length of cluster bitmap.
```

```
                 07  11 0B00 11716       BRB   30$                         ; Jump to common code.
  0000226C'EF    00  D0 0B02 11717 20$:  MOVL  #LOCAL_MAP_LEN,MAP_LEN      ; Save length of local bitmap.
             1857'   30 0B09 11718 30$:  BSBW  FIND_ENTRY                  ; Search for any field being displayed.
              15 50  E9 0B0C 11719       BLBC  R0,40$                      ; Branch if none found.
              0C AA  D1 0B0F 11720       CMPL  FD$L_CLASS(R10),-           ; Is the class for the field we just
              0C A3     0B12 11721             FD$L_CLASS(R3)              ;  found the same as the field we are
                        0B14 11722                                         ;  adding to the display?
                 F3  12 0B14 11723       BNEQ  30$                         ; If not go find another field.
           5A    53  D1 0B16 11724       CMPL  R3,R10                      ; Is it the same field?
                 EE  13 0B19 11725       BEQL  30$                         ; If so go find another field.
  62    00000547'EF  CA 0B1B 11726       BICL  KEY_MSK,(R2)                ; Clear the bit, turn off the field.
                 15  11 0B22 11727       BRB   99$                         ; Branch to common code.
              0C A3  CA 0B24 11728 40$:  BICL  FD$L_CLASS(R3),-            ; No other fields in this class found,
            F4D7 CF     0B27 11729             CLASS_SELECT                ;  so turn off the class, not the field.
                 00  E1 0B2A 11730       BBC   #CLS$V_CLUB,-               ; Did we just turn off the cluster
           07 0C A3     0B2C 11731             FD$L_CLASS(R3),50$          ;  class?
  00002254'EF    07  D0 0B2F 11732       MOVL  #7,FIRST_DATA_LINE          ; If so, set first line to be scrolled.
           50    01  D0 0B36 11733 50$:  MOVL  #SS$_NORMAL,R0              ; Return success.
                     04 0B39 11734 99$:  RET
                        0B3A 11735
                        0B3A 11736
```

```
                                        0B3A        .SBTTL          -       LOOKUP_FIELD
                                        0B3A   ;++
                                        0B3A   ;
                                        0B3A   ; FUNCTIONAL DESCRIPTION:
                                        0B3A   ;
                                        0B3A   ;           This routine is used to parse the field name by calling
                                        0B3A   ;       LIB$LOOKUP_KEY using each of the keyword tables containing the field
                                        0B3A   ;       names.  The keyword value is converted into an address and mask for the
                                        0B3A   ;       bit within the appropriate bitmap and the address of the descriptor for
                                        0B3A   ;       the field.
                                        0B3A   ;
                                        0B3A   ; CALLING SEQUENCE:
                                        0B3A   ;
                                        0B3A   ;       CALLS   #0,LOOKUP_FIELD
                                        0B3A   ;
                                        0B3A   ; CALLED BY:
                                        0B3A   ;
                                        0B3A   ;       ADD_FIELD
                                        0B3A   ;       REMOVE_FIELD
                                        0B3A   ;
                                        0B3A   ; INPUTS:
                                        0B3A   ;
                                        0B3A   ; IMPLICIT INPUTS:
                                        0B3A   ;
                                        0B3A   ;       Selected fieldname stored at address TPARSE_BLK+TPA$L_TOKENCNT.
                                        0B3A   ;       Address of keyword table stored at KEYTAB_ADRS.
                                        0B3A   ;       Address of bitmap stored at BITMAP_ADRS.
                                        0B3A   ;       Address of address table stored at ADRS_TABLE.
                                        0B3A   ;
                                        0B3A   ; OUTPUTS:
                                        0B3A   ;
                                        0B3A   ;       KEY_OFF  = Address of longword in the bitmap that contains the bit
                                        0B3A   ;                  corresponding to the field selected.
                                        0B3A   ;       KEY_MSK  = Mask of the bit within KEY_OFF corresponding to the field.
                                        0B3A   ;       KEY_DSCR = Address of the field descriptor for the selected field.
                                        0B3A   ;
                                        0B3A   ; RETURN STATUS:
                                        0B3A   ;
                                        0B3A   ;       SS$_NORMAL or failed status from LIB$LOOKUP_KEY returned in R0.
                                        0B3A   ;
                                        0B3A   ;--
                                        0B3A
                                 003C   0B3A   .ENTRY  LOOKUP_FIELD,^M<R2,R3,R4,R5>
         53   00002274'EF          D0   0B3C           MOVL    KEYTAB_ADRS,R3            ; Get address of keyword table.
         54   00002268'EF          D0   0B43           MOVL    BITMAP_ADRS,R4            ; Get address of bitmap.
         55   00002264'EF          D0   0B4A           MOVL    ADRS_TABLE,R5             ; Get address of address table.
              0000053F'EF          DF   0B51           PUSHAL  KEY_VAL                   ; Push address to receive value.
                           53      DD   0B57           PUSHL   R3                        ; Push keyword table address.
                                        0B59                                             ; Push address of descriptor for the
                                        0B59                                             ;  remainder of the string.
            000004D7'EF            DF   0B59           PUSHAL  TPARSE_BLK+TPA$L_TOKENCNT
      00000000'GF          03      FB   0B5F           CALLS   #3,G^LIB$LOOKUP_KEY       ; Check for valid field name in the
                                        0B66                                             ;  current report.
                        2D 50      E8   0B66           BLBS    R0,30$                    ; Branch if field name is valid in the
                                        0B69                                             ;  current report.
                                        0B69   ;
                                        0B69   ; If the field lookup fails, then determine if it is a valid field name,
```

```
                            0B69 11795 ; but simply can't be displayed in the current report.  If so, then pass
                            0B69 11796 ; the additional information on to the user.
                            0B69 11797 ;
                  00    E1  0B69 11798         BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
            08 F492 CF      0B6B 11799                 CURRENT_REPORT,10$      ;  cluster report.
          00000000'EF   DF  0B6F 11800         PUSHAL  LOCAL_KEYTAB            ; Lookup field in the LOCAL keyword
                            0B75 11801                                         ;  table.
                  06    11  0B75 11802         BRB     20$                     ; Jump to common code.
          00000000'EF   DF  0B77 11803 10$:    PUSHAL  CLUSTER_KEYTAB          ; Check for valid field name.
          000004D7'EF   DF  0B7D 11804 20$:    PUSHAL  TPARSE_BLK+TPA$L_TOKENCNT
     00000000'GF   02   FB  0B83 11805         CALLS   #2,G^LIB$LOOKUP_KEY     ; Check for valid field name.
               3C  50   E9  0B8A 11806         BLBC    R0,99$                  ; Branch on error.
     50    00000000'8F  D0  0B8D 11807         MOVL    #SHOW$_NOAVAIL,R0       ; "Field not available in this report".
                  33    11  0B94 11808         BRB     99$                     ; Exit with status.
     52    0000053F'EF  D0  0B96 11809 30$:    MOVL    KEY_VAL,R2              ; Get the keyword value in R2,R3.
                  53    D4  0B9D 11810         CLRL    R3
   53    52    52   20  7B  0B9F 11811         EDIV    #32,R2,R2,R3            ; Convert the keyword value into an
                            0BA4 11812                                         ;  offset into a field bitmap.
                            0BA4 11813                                         ;  R2 gets the longword offset,
                            0BA4 11814                                         ;  R3 gets the bit offset.
   00000543'EF   6442   DE  0BA4 11815         MOVAL   (R4)[R2],KEY_OFF        ; Get address of longword containing
                            0BAC 11816                                         ;  the bit and store in KEY_OFF.
                            0BAC 11817 ;
                            0BAC 11818 ; Convert the bit position in R3 to a mask.
                            0BAC 11819 ;
               54  01   D0  0BAC 11820         MOVL    #1,R4                   ; Start with mask for bit 0.
 00000547'EF   54  53   78  0BAF 11821         ASHL    R3,R4,KEY_MSK           ; Shift left by the number a bits in R3
                            0BB7 11822                                         ;  and save in KEY_MSK.
                            0BB7 11823
     54    0000053F'EF  D0  0BB7 11824         MOVL    KEY_VAL,R4
   0000054B'EF   6544   D0  0BBE 11825         MOVL    (R5)[R4],KEY_DSCR       ; Get address of field descriptor for
                            0BC6 11826                                         ;  this keyword and save in KEY_DSCR.
               50  01   D0  0BC6 11827         MOVL    #SS$_NORMAL,R0          ; Indicate success.
                        04  0BC9 11828 99$:    RET
                            0BCA 11829
                            0BCA 11830
```

```
                                 0BCA         .SBTTL          -       ADD_ALL_FIELDS
                                 0BCA  ;++
                                 0BCA  ;
                                 0BCA  ; FUNCTIONAL DESCRIPTION:
                                 0BCA  ;
                                 0BCA  ;      This routine is used to turn on all the fields in a particular class
                                 0BCA  ;      when the /ALL qualifier is used with the ADD class-name command.
                                 0BCA  ;
                                 0BCA  ; CALLING SEQUENCE:
                                 0BCA  ;
                                 0BCA  ;      BRB     ADD_ALL_FIELDS
                                 0BCA  ;
                                 0BCA  ; CALLED BY:
                                 0BCA  ;
                                 0BCA  ;      READ_W_PROMPT
                                 0BCA  ;
                                 0BCA  ; INPUTS:
                                 0BCA  ;
                                 0BCA  ;      Class select bit for the class to have all fields turned on at
                                 0BCA  ;      TPARSE_BLK+TPA$L_PARAM.
                                 0BCA  ;
                                 0BCA  ; OUTPUTS:
                                 0BCA  ;
                                 0BCA  ;      Bits corresponding to the the fields in the selected class set in the
                                 0BCA  ;      bitmap.
                                 0BCA  ;      Bit corresponding to the selected class set at CLASS_SELECT.
                                 0BCA  ;
                                 0BCA  ;
                                 0BCA  ;--
                                 0BCA
                           06E0  0BCA  .ENTRY  ADD_ALL_FIELDS,^M<R5,R6,R7,R9,R10>
    00000C2D'EF     00       FB  0BCC          CALLS   #0,INCLUDE_CLASS        ; Make sure the class is add to the
                                 0BD3                                          ;  display.
 56   00002264'EF            D0  0BD3          MOVL    ADRS_TABLE,R6           ; Get the address of the address table
                                 0BDA                                          ;  for the field descriptors.
 57   00002268'EF            D0  0BDA          MOVL    BITMAP_ADRS,R7          ; Get the address of field descriptor
                                 0BE1                                          ;  bitmap for the current report.
               59   01       D0  0BE1          MOVL    #1,R9                   ; Start with bit 0.
                    00       E1  0BE4          BBC     #RPT$V_CLSTR,-          ; Branch if we're not displaying the
            09 F417 CF           0BE6                  CURRENT_REPORT,10$      ;  cluster report.
 55   00000000'EF            D0  0BEA          MOVL    CLUSTER_KEYTAB,R5       ; Get the number of cluster keyword
                                 0BF1                                          ;  vectors.
                    07       11  0BF1          BRB     15$                     ; Jump to common code.
 55   00000000'EF            D0  0BF3  10$:    MOVL    LOCAL_KEYTAB,R5         ; Get the number of local keyword
                                 0BFA                                          ;  vectors.
  55     55     FF  8F       78  0BFA  15$:    ASHL    #-1,R5,R5               ; Get the number of fields to check.
                    5A       D4  0BFF  16$:    CLRL    R10                     ; Dividend must be a quadword.
5A     51     59    20       7B  0C01          EDIV    #32,R9,R1,R10           ; Calculate the number of bits left to
       5A     20    5A       C3  0C06          SUBL3   R10,#32,R10             ;  scan in the longword.
59     67     5A    59       EB  0C0A          FFC     R9,R10,(R7),R9          ; Scan for a clear bit.
                    EE       13  0C0F          BEQLU   16$                     ; If not, scan next longword.
              55    59       D1  0C11          CMPL    R9,R5                   ; Have we checked all fields?
                    16       13  0C14          BEQL    99$                     ; Branch if so.
                                 0C16
          5A      6649       D0  0C16  20$:    MOVL    (R6)[R9],R10            ; Get the address of the field
                                 0C1A                                          ;  descriptor.
                 0C AA       D3  0C1A          BITL    FD$L_CLASS(R10),-       ; Is the field in the desired class?
```

```
000004E7'EF      0C1D 11889              TPARSE_BLK+TPA$L_PARAM  ;
         04   13 0C22 11890         BEQL  25$                    ; If not go find another field.
   00 67 59   E2 0C24 11891         BBSS  R9,(R7),25$            ; Otherwise, turn on the field.
         59   D6 0C28 11892 25$:    INCL  R9                     ; Get number of next bit to test.
         D3   11 0C2A 11893         BRB   16$                    ; Loop for all fields.
              04 0C2C 11894 99$:    RET
                 0C2D 11895
                 0C2D 11896
```

```
                                  0C2D        .SBTTL          -               INCLUDE_CLASS
                                  0C2D  ;++
                                  0C2D  ;
                                  0C2D  ; FUNCTIONAL DESCRIPTION:
                                  0C2D  ;
                                  0C2D  ;         This routine is used to add a class to the display.  It simply sets the
                                  0C2D  ;         bit that corresponds to the desired class in location CLASS_SELECT.
                                  0C2D  ;
                                  0C2D  ; CALLING SEQUENCE:
                                  0C2D  ;
                                  0C2D  ;         CALLS   #0,INCLUDE_CLASS
                                  0C2D  ;
                                  0C2D  ; CALLED BY:
                                  0C2D  ;
                                  0C2D  ;         READ_W_PROMPT
                                  0C2D  ;         ADD_ALL_FIELDS
                                  0C2D  ;
                                  0C2D  ; INPUTS:
                                  0C2D  ;
                                  0C2D  ;         Mask of the bit to be set at address TPARSE_BLK+TPA$L_PARAM.
                                  0C2D  ;
                                  0C2D  ; OUTPUTS:
                                  0C2D  ;
                                  0C2D  ;         Bit corresponding to the desired class set at CLASS_SELECT.
                                  0C2D  ;
                                  0C2D  ; RETURN STATUS:
                                  0C2D  ;
                                  0C2D  ;         SS$_NORMAL if class can be added to the current report, otherwise
                                  0C2D  ;         SHOW$_CACLS if the class can't be added to the current report.
                                  0C2D  ;
                                  0C2D  ;--
                                  0C2D
                            0000  0C2D  .ENTRY  INCLUDE_CLASS,^M<>
                                  0C2F
                     00       E1  0C2F          BBC     #RPT$V_CLSTR,-                  ; Branch if we're not displaying the
             0F F3CC CF           0C31                  CURRENT_REPORT,10$              ;  cluster report.
            00000180 8F       D3  0C35          BITL    #CLS$M_LPRT!CLS$M_ERR,-         ; Is the desired class in the
            000004E7'EF           0C3B                  TPARSE_BLK+TPA$L_PARAM          ;  LOCAL_PORTS report?
                     2C       12  0C40          BNEQ    98$                             ; Error if so.
                     0D       11  0C42          BRB     20$                             ; Branch to common code.
                                  0C44
            0000007F 8F       D3  0C44  10$:    BITL    #CLS$M_ANY_CLUS,-               ; Is the desired class in the CLUSTER
            000004E7'EF           0C4A                  TPARSE_BLK+TPA$L_PARAM          ;  report?
                     1D       12  0C4F          BNEQ    98$                             ; Error if so.
                                  0C51
            000004E7'EF       C8  0C51  20$:    BISL    TPARSE_BLK+TPA$L_PARAM,-        ; Set the bit corresponding to the
                   F3A7 CF        0C57                  CLASS_SELECT                    ;  desired class.
                     00       E1  0C5A          BBC     #CLS$V_CLUB,-                   ; Branch if desired class isn't
            000004E7'EF           0C5C                  TPARSE_BLK+TPA$L_PARAM,-        ;  CLUSTER
                     07           0C61                  30$
   00002254'EF       0E       D0  0C62          MOVL    #14,FIRST_DATA_LINE             ; Set first line to be scrolled.
            50       01       D0  0C69  30$:    MOVL    #SS$_NORMAL,R0                  ; Indicate success.
                     12       11  0C6C          BRB     99$                             ; Branch to common exit.
                                  0C6E
000004A9'EF   00000495'EF     DE  0C6E  98$:    MOVAL   TKN_DSCR,ERR_MSGVEC+12          ; Save descriptor for class name.
         50   00000000'8F     D0  0C79          MOVL    #SHOW$_CACLS,R0                 ; "Can't add the class to this report"
                              04  0C80  99$:    RET
```

```
0C81 11955
0C81 11956
```

```
                                  0C81          .SBTTL          -       EXCLUDE_CLASS
                                  0C81  ;++
                                  0C81  ;
                                  0C81  ; FUNCTIONAL DESCRIPTION:
                                  0C81  ;
                                  0C81  ;         This routine is used to remove a class from the display.  It simply
                                  0C81  ;         clears the bit that corresponds to the desired class in location
                                  0C81  ;         CLASS_SELECT.
                                  0C81  ;
                                  0C81  ; CALLING SEQUENCE:
                                  0C81  ;
                                  0C81  ;         CALLS   #0,EXCLUDE_CLASS
                                  0C81  ;
                                  0C81  ; CALLED BY:
                                  0C81  ;
                                  0C81  ;         READ_W_PROMPT
                                  0C81  ;
                                  0C81  ; INPUTS:
                                  0C81  ;
                                  0C81  ;         Mask of the bit to be cleared at address TPARSE_BLK+TPA$L_PARAM.
                                  0C81  ;
                                  0C81  ; OUTPUTS:
                                  0C81  ;
                                  0C81  ;         Bit corresponding to the desired class cleared at CLASS_SELECT.
                                  0C81  ;
                                  0C81  ; RETURN STATUS:
                                  0C81  ;
                                  0C81  ;         SS$_NORMAL if class can be removed from the current report, otherwise
                                  0C81  ;         SHOW$_CRCLS if the class can't be removvrd from the current report.
                                  0C81  ;
                                  0C81  ;--
                                  0C81
                           0000   0C81  .ENTRY  EXCLUDE_CLASS,^M<>
                                  0C83
                     00     E1    0C83          BBC     #RPT$V_CLSTR,-               ; Branch if we're not displaying the
             0F F378 CF           0C85                  CURRENT_REPORT,10$           ;  cluster report.
            00000180 8F     D3    0C89          BITL    #CLS$M_[PRT!CLS$M_ERR,-      ; Is the desired class in the
           000004E7'EF            0C8F                  TPARSE_BLK+TPA$L_PARAM       ;  LOCAL_PORTS report?
                     2C     12    0C94          BNEQ    98$                          ; Error if so.
                     0D     11    0C96          BRB     20$                          ; Branch to common code.
                                  0C98
            0000007F 8F     D3    0C98  10$:    BITL    #CLS$M_ANY_CLUS,-            ; Is the desired class in the CLUSTER
           000004E7'EF            0C9E                  TPARSE_BLK+TPA$L_PARAM       ;  report?
                     1D     12    0CA3          BNEQ    98$                          ; Error if so.
                                  0CA5
           000004E7'EF      CA    0CA5  20$:    BICL    TPARSE_BLK+TPA$L_PARAM,-     ; Clear the bit corresponding to the
                F353 CF           0CAB                  CLASS_SELECT                 ;  desired class.
                     00     E1    0CAE          BBC     #CLS$V_CLUB,-                ; Branch if desired class isn't
           000004E7'EF            0CB0                  TPARSE_BLK+TPA$L_PARAM,-     ;  CLUSTER
                     07           0CB5                  30$
 00002254'EF         07     D0    0CB6          MOVL    #7,FIRST_DATA_LINE           ; Set first line to be scrolled.
          50         01     D0    0CBD  30$:    MOVL    #SS$_NORMAL,R0               ; Indicate success.
                     12     11    0CC0          BRB     99$                          ; Branch to common exit.
                                  0CC2
 000004A9'EF  00000495'EF   DE    0CC2  98$:    MOVAL   TKN_DSCR,ERR_MSGVEC+12       ; Save descriptor for class name.
          50  00000000'8F   D0    0CCD          MOVL    #SHOW$_CRCLS,R0              ; Assume we can't remove the class from
                                  0CD4                                               ;  this report.
```

```
04  0CD4 12015 99$:    RET
    0CD5 12016
    0CD5 12017
```

```
                                        0CD5          .SBTTL          -          ADD_ID
                                        0CD5
                                        0CD5  ;++
                                        0CD5  ;
                                        0CD5  ; FUNCTIONAL DESCRIPTION:
                                        0CD5  ;
                                        0CD5  ;       This routine adds a system ID to the ID list for systems that will not
                                        0CD5  ;       be included in the display.
                                        0CD5  ;
                                        0CD5  ;
                                        0CD5  ; CALLING SEQUENCE:
                                        0CD5  ;
                                        0CD5  ;       CALLS   #0,ADD_ID
                                        0CD5  ;
                                        0CD5  ; CALLED BY:
                                        0CD5  ;
                                        0CD5  ;       READ_W_PROMPT
                                        0CD5  ;
                                        0CD5  ; INPUTS:
                                        0CD5  ;
                                        0CD5  ;       TKN_DSCR Contains the descriptor for the string which specifies the
                                        0CD5  ;       ID of the system to add to the list.
                                        0CD5  ;
                                        0CD5  ; OUTPUTS:
                                        0CD5  ;
                                        0CD5  ;       Six byte hex ID number is stored in the list of ID numbers for systems
                                        0CD5  ;       to be removed from the display.
                                        0CD5  ;
                                        0CD5  ;       The number of entries in the list is incremented (ID_COUNT).
                                        0CD5  ;
                                        0CD5  ; RETURN STATUS:
                                        0CD5  ;
                                        0CD5  ;       SS$_NORMAL or failed status returned in R0.
                                        0CD5  ;
                                        0CD5  ;--
                                        0CD5
                                  0000  0CD5  .ENTRY  ADD_ID,^M<>
                                        0CD7
                      00000E5C'EF   16  0CD7          JSB     CONVERT_TO_HEX          ; Convert system ID string to hex value.
                            09 50   E9  0CDD          BLBC    R0,98$                  ; Branch on error.
                      00000DA0'EF   16  0CE0          JSB     ADD_TO_LIST             ; Add the system ID to the list.
                            12 50   E8  0CE6          BLBS    R0,99$                  ; Exit on success.
               50     00000000'8F   D0  0CE9  98$:    MOVL    #SHOW$_IVREMID,R0       ; "Can't remove system from the display"
      000004A9'EF     00000495'EF   DE  0CF0          MOVAL   TKN_DSCR,ERR_MSGVEC+12  ; Save descriptor for class name.
                                    04  0CFB  99$:    RET                             ; Return with status.
                                        0CFC
                                        0CFC
```

```
                              0CFC        .SBTTL          -       REM_NODE
                              0CFC
                              0CFC ;++
                              0CFC ;
                              0CFC ; FUNCTIONAL DESCRIPTION:
                              0CFC ;
                              0CFC ;         This routine removes a system from the display by converting the node
                              0CFC ;         name of the system to the system's ID and then adding the system's ID
                              0CFC ;         to the ID list for systems that will not be included in the display.
                              0CFC ;
                              0CFC ;
                              0CFC ; CALLING SEQUENCE:
                              0CFC ;
                              0CFC ;         CALLS   #0,REM_NODE
                              0CFC ;
                              0CFC ; CALLED BY:
                              0CFC ;
                              0CFC ;         READ_W_PROMPT
                              0CFC ;
                              0CFC ; INPUTS:
                              0CFC ;
                              0CFC ;         TKN_DSCR Contains the descriptor for the string which specifies the
                              0CFC ;         node name of the system to remove from the display.
                              0CFC ;
                              0CFC ; OUTPUTS:
                              0CFC ;
                              0CFC ;         Six byte hex ID number is stored in the list of ID numbers for systems
                              0CFC ;         to be removed from the display.
                              0CFC ;
                              0CFC ;         The number of entries in the list is incremented (ID_COUNT).
                              0CFC ;
                              0CFC ; RETURN STATUS:
                              0CFC ;
                              0CFC ;         SS$_NORMAL or failed status returned in R0.
                              0CFC ;
                              0CFC ;--
                              0CFC
                        0000  0CFC .ENTRY  REM_NODE,^M<>
                              0CFE
         00000D5B'EF      16  0CFE         JSB     FIND_NODE               ; Find the ID of the system that
                              0D04                                         ;  coresponds to the specified node
                              0D04                                         ;  name.
               09 50      E9  0D04         BLBC    R0,98$                  ; Branch if not found.
         00000DA0'EF      16  0D07 10$:    JSB     ADD_TO_LIST             ; Add the system ID to the list.
               07 50      E8  0D0D         BLBS    R0,99$                  ; Exit on success.
                              0D10
   50    00000000'8F      D0  0D10 98$:    MOVL    #SHOW$_IVREMNODE,R0     ; Indicate specified system can not be
                              0D17                                         ;  removed.
                          04  0D17 99$:    RET                             ; Return with status.
                              0D18
                              0D18
```

```
                                 0D18          .SBTTL          -       REM_ID
                                 0D18
                                 0D18  ;++
                                 0D18  ;
                                 0D18  ; FUNCTIONAL DESCRIPTION:
                                 0D18  ;
                                 0D18  ;       This routine removes adds a system to the display by removing it's
                                 0D18  ;       system ID from the ID list of systems that will not be included in
                                 0D18  ;       the display.
                                 0D18  ;
                                 0D18  ;
                                 0D18  ; CALLING SEQUENCE:
                                 0D18  ;
                                 0D18  ;       CALLS   #0,REM_ID
                                 0D18  ;
                                 0D18  ; CALLED BY:
                                 0D18  ;
                                 0D18  ;       READ_W_PROMPT
                                 0D18  ;
                                 0D18  ; INPUTS:
                                 0D18  ;
                                 0D18  ;       TKN_DSCR Contains the descriptor for the string which specifies the
                                 0D18  ;       ID of the system to remove from the list.
                                 0D18  ;
                                 0D18  ; OUTPUTS:
                                 0D18  ;
                                 0D18  ;       The six byte hex ID number stored in the list of ID numbers for systems
                                 0D18  ;       to be removed from the display is cleared.
                                 0D18  ;
                                 0D18  ;       The number of entries in the list is decremented (ID_COUNT).
                                 0D18  ;
                                 0D18  ; RETURN STATUS:
                                 0D18  ;
                                 0D18  ;       SS$_NORMAL or failed status returned in R0.
                                 0D18  ;
                                 0D18  ;--
                                 0D18
                           0000  0D18  .ENTRY  REM_ID,^M<>
                                 0D1A
             00000E5C'EF    16   0D1A          JSB     CONVERT_TO_HEX          ; Convert system ID string to hex value.
                   09 50    E9   0D20          BLBC    R0,98$                  ; Branch on error.
             00000DFE'EF    16   0D23          JSB     REMOVE_FROM_LIST        ; Remove the system ID from
                                 0D29                                          ;  SYS_ID_LIST.
                   12 50    E8   0D29          BLBS    R0,99$                  ; Exit on success.
       50    00000000'8F    D0   0D2C  98$:    MOVL    #SHOW$_IVADDID,R0       ; "Can't add system to the display"
000004A9'EF  00000495'EF    DE   0D33          MOVAL   TKN_DSCR,ERR_MSGVEC+12  ; Save descriptor for system ID.
                            04   0D3E  99$:    RET                             ; Return with status.
                                 0D3F
                                 0D3F
```

```
                         0D3F          .SBTTL          -       ADD_NODE
                         0D3F
                         0D3F  ;++
                         0D3F  ;
                         0D3F  ; FUNCTIONAL DESCRIPTION:
                         0D3F  ;
                         0D3F  ;       This routine adds a system to the display by converting the node
                         0D3F  ;       name specified into the system ID and removing the system ID
                         0D3F  ;       from the ID list for systems that will not be included in the display.
                         0D3F  ;
                         0D3F  ;
                         0D3F  ; CALLING SEQUENCE:
                         0D3F  ;
                         0D3F  ;       CALLS   #0,ADD_NODE
                         0D3F  ;
                         0D3F  ; CALLED BY:
                         0D3F  ;
                         0D3F  ;       READ_W_PROMPT
                         0D3F  ;
                         0D3F  ; INPUTS:
                         0D3F  ;
                         0D3F  ;       TKN_DSCR Contains the descriptor for the string which specifies the
                         0D3F  ;       node name of the system to remove from the list.
                         0D3F  ;
                         0D3F  ; OUTPUTS:
                         0D3F  ;
                         0D3F  ;       The six byte hex ID number stored in the list of ID numbers for systems
                         0D3F  ;       to be removed from the display is cleared.
                         0D3F  ;
                         0D3F  ;       The number of entries in the list is decremented (ID_COUNT).
                         0D3F  ;
                         0D3F  ; RETURN STATUS:
                         0D3F  ;
                         0D3F  ;       R0 =    SS$_NORMAL
                         0D3F  ;               SHOW$_IVADDNODE if node is unknown or already being displayed.
                         0D3F  ;
                         0D3F  ;--
                         0D3F
                   0000  0D3F  .ENTRY  ADD_NODE,^M<>
                         0D41
      00000D5B'EF    16  0D41          JSB     FIND_NODE               ; Find the ID of the system that
                         0D47                                          ;   coresponds to the specified node
                         0D47                                          ;   name.
            09 50    E9  0D47          BLBC    R0,98$                  ; Branch if not found.
      00000DFE'EF    16  0D4A          JSB     REMOVE_FROM_LIST        ; Remove the system ID from the list of
                         0D50                                          ;   systems removed from the display.
            07 50    E8  0D50          BLBS    R0,99$                  ; Exit on success.
50    00000000'8F    D0  0D53  98$:    MOVL    #SHOW$_IVADDNODE,R0     ; Indicate specified node can not be
                         0D5A                                          ;   added.
                     04  0D5A  99$:    RET                             ; Return with status.
                         0D5B
                         0D5B
```

```
                              0D5B                .SBTTL          -       FIND_NODE
                              0D5B
                              0D5B   ;++
                              0D5B   ;
                              0D5B   ; FUNCTIONAL DESCRIPTION:
                              0D5B   ;
                              0D5B   ;       This routine is used to convert the node name specified by the
                              0D5B   ;       REMOVE or ADD SYSTEMS/NODE= commands into a system ID.  The
                              0D5B   ;       system blocks in SB buffer are scanned for a node name equal to
                              0D5B   ;       the node name specified.  If a match is found, then the system
                              0D5B   ;       ID of the matching system block is taken from the system block
                              0D5B   ;       and stored in ID_RESULT.  The system ID can then be used to add
                              0D5B   ;       or remove the system ID from SYS_ID_LIST.
                              0D5B   ;
                              0D5B   ; CALLING SEQUENCE:
                              0D5B   ;
                              0D5B   ;       JSB     FIND_NODE
                              0D5B   ;
                              0D5B   ; CALLED BY:
                              0D5B   ;
                              0D5B   ;       REM_NODE
                              0D5B   ;       ADD_NODE
                              0D5B   ;
                              0D5B   ; INPUTS:
                              0D5B   ;
                              0D5B   ;       TKN_DSCR contains a descriptor for the node name string specified.
                              0D5B   ;
                              0D5B   ; OUTPUTS:
                              0D5B   ;
                              0D5B   ;       System ID for the node specified stored in ID_RESULT.
                              0D5B   ;
                              0D5B   ; RETURN STATUS:
                              0D5B   ;
                              0D5B   ;       R0 = SS$_NORMAL if a system block and system ID were found that match
                              0D5B   ;            the node name.
                              0D5B   ;       R0 = 0 if not found.
                              0D5B   ;
                              0D5B   ; REGISTER USAGE:
                              0D5B   ;
                              0D5B   ;       R2  - Used by CMPC.
                              0D5B   ;       R3  - Used by CMPC.
                              0D5B   ;       R10 - Points to the base of a system block in SB buffer.
                              0D5B   ;
                              0D5B   ;--
                              0D5B
                              0D5B   FIND_NODE:
              040C 8F    BB   0D5B           PUSHR   #^M<R2,R3,R10>
     5A  000003ED'EF     D0   0D5F           MOVL    VAR_BUF,R10             ; Get the address of the system block
                              0D66                                           ;  buffer.
         00000495'EF     91   0D66   10$:    CMPB    TKN_DSCR,-              ; Is the number of characters specified
                44 AA         0D6C                   SB$T_NODENAME(R10)      ;  the same as the length of the node
                              0D6E                                           ;  name in this system block?
                   0F    12   0D6E           BNEQ    20$                     ; If not this can't be the right system
                              0D70                                           ;  block.
00000499'FF  00000495'EF  29  0D70           CMPC3   TKN_DSCR,@TKN_DSCR+4,-  ; Compare the nodename specified with
                45 AA         0D7B                   SB$T_NODENAME+1(R10)    ;  the nodename in the system block.
                   09    13   0D7D           BEQL    30$                     ; Branch if they're the same.
```

```
        5A    6A   D0  0D7F 12279 20$:  MOVL    (R10),R10              ; Get the address of the next system
                       0D82 12280                                      ;  block.
              E2   12  0D82 12281       BNEQ    10$                    ; Loop of there's more SB's.
              50   D4  0D84 12282       CLRL    R0                     ; Otherwise, indicate failure,
              13   11  0D86 12283       BRB     99$                    ;  and return with error status.
           18 AA   D0  0D88 12284 30$:  MOVL    SB$B_SYSTEMID(R10),-   ; Store the system ID for the matching
    000022DC'EF        0D8B 12285       ID_RESULT                      ;  system block in ID_RESULT.
           1C AA   B0  0D90 12286       MOVW    SB$B_SYSTEMID+4(R10),- ;
    000022E0'EF        0D93 12287       ID_RESULT+4                    ;
        50    01   D0  0D98 12288       MOVL    #SS$_NORMAL,R0         ; Indicate success.
      040C    8F   BA  0D9B 12289 99$:  POPR    #^M<R2,R3,R10>
                   05  0D9F 12290       RSB
                       0DA0 12291
                       0DA0 12292
```

```
                         0DA0         .SBTTL          -       ADD_TO_LIST
                         0DA0
                         0DA0  ;++
                         0DA0  ;
                         0DA0  ; FUNCTIONAL DESCRIPTION:
                         0DA0  ;
                         0DA0  ;        The routine disables a system from being displayed by adding it's
                         0DA0  ;        system ID to the list of systems not to be displayed.  It searches
                         0DA0  ;        the list at SYS_ID_LIST to make sure that the system id is not
                         0DA0  ;        already in the list.  If it isn't, then it inserts the system ID
                         0DA0  ;        in the first empty entry.
                         0DA0  ;
                         0DA0  ; CALLING SEQUENCE:
                         0DA0  ;
                         0DA0  ;        JSB     ADD_TO_LIST
                         0DA0  ;
                         0DA0  ; CALLED BY:
                         0DA0  ;
                         0DA0  ;        ADD_ID
                         0DA0  ;        REM_NODE
                         0DA0  ;
                         0DA0  ; INPUTS:
                         0DA0  ;
                         0DA0  ;        SYS_ID_LIST contains a list of system ID's for systems removed from
                         0DA0  ;        the display.
                         0DA0  ;        ID_RESULT contains the ID of the system to remove add to the list.
                         0DA0  ;        ID_COUNT contains the number of system entered in the list.
                         0DA0  ;
                         0DA0  ; OUTPUTS:
                         0DA0  ;
                         0DA0  ;        System ID for entered in SYS_ID_LIST.
                         0DA0  ;        ID_COUNT updated.
                         0DA0  ;
                         0DA0  ; RETURN STATUS:
                         0DA0  ;
                         0DA0  ;        R0 = SS$_NORMAL or failed status if the entry is already in the
                         0DA0  ;             list or if there is no more room in the list.
                         0DA0  ;
                         0DA0  ; REGISTER USAGE:
                         0DA0  ;
                         0DA0  ;        R2  - Used as an index for comparing the six bytes of the system ID.
                         0DA0  ;        R4  - Used as a pointer into the SYS_ID_LIST.
                         0DA0  ;        R5  - Used to count down the total number of possible entries.
                         0DA0  ;        R6  - Used to save the address of the first empty entry found.
                         0DA0  ;
                         0DA0  ;--
                         0DA0
                         0DA0  ADD_TO_LIST:
        0074 8F     BB   0DA0         PUSHR   #^M<R2,R4,R5,R6>
             50     D4   0DA4         CLRL    R0                      ; Assume failure.
 54   00002278'EF   DE   0DA6         MOVAL   SYS_ID_LIST,R4          ; Get the address of the list.
             56     D4   0DAD         CLRL    R6                      ; Assume no empty entry.
       55    10     D0   0DAF         MOVL    #ID_LIST_ENTRYS,R5      ; Get the number of entries to search.
       52    02     D0   0DB2         MOVL    #2,R2                   ; R2 = word index.
 64   000022DC'EF   D1   0DB5  10$:   CMPL    ID_RESULT,(R4)          ; Compare the specified result with
                         0DBC                                         ;  the entry in the list.
             0A     12   0DBC         BNEQ    20$                     ; Branch if no match.
```

```
        6442  000022E0'EF   B1  0DBE 12351          CMPW    ID_RESULT+4,(R4)[R2]    ; Compare all 6 bytes.
                       31   13  0DC6 12352          BEQL    99$                     ; Branch if the system ID is already in
                                0DC8 12353                                          ;  the list.
                       56   D5  0DC8 12354 20$:     TSTL    R6                      ; Have we saved the address of the
                                0DCA 12355                                          ;  empty entry in R6?
                       0C   12  0DCA 12356          BNEQ    30$                     ; Branch if so.
                       64   D5  0DCC 12357          TSTL    (R4)                    ; Is this entry empty?
                       08   12  0DCE 12358          BNEQ    30$                     ; Branch if not.
                     6442   B5  0DD0 12359          TSTW    (R4)[R2]                ; Check all 6 bytes.
                       03   12  0DD3 12360          BNEQ    30$                     ; Branch if entry isn't empty.
                  56   54   D0  0DD5 12361          MOVL    R4,R6                   ; Save the address of the empty entry.
                  54   06   C0  0DD8 12362 30$:     ADDL    #6,R4                   ; Point to next entry.
                    D7 55   F5  0DDB 12363          SOBGTR  R5,10$                  ; Loop for entire list.
                       56   D5  0DDE 12364          TSTL    R6                      ; Did we find an empty entry.
                       17   13  0DE0 12365          BEQL    99$                     ; Branch if not.
          86  000022DC'EF   D0  0DE2 12366          MOVL    ID_RESULT,(R6)+         ; Otherwise, save the system ID in the
          66  000022E0'EF   B0  0DE9 12367          MOVW    ID_RESULT+4,(R6)        ;  list.
              000022D8'EF   D6  0DF0 12368          INCL    ID_COUNT                ; Increase count of entries in the
                                0DF6 12369                                          ;  list.
                  50   01   D0  0DF6 12370          MOVL    #SS$_NORMAL,R0          ; Indicate success.
                0074 8F     BA  0DF9 12371 99$:     POPR    #^M<R2,R4,R5,R6>
                            05  0DFD 12372          RSB
                                0DFE 12373
                                0DFE 12374
```

```
                          0DFE        .SBTTL          -       REMOVE_FROM_LIST
                          0DFE
                          0DFE  ;++
                          0DFE  ;
                          0DFE  ; FUNCTIONAL DESCRIPTION:
                          0DFE  ;
                          0DFE  ;       This routine clears the entry for a system in the ID list of systems
                          0DFE  ;       that are not to be included in the display thereby adding the system
                          0DFE  ;       to the display.
                          0DFE  ;
                          0DFE  ; CALLING SEQUENCE:
                          0DFE  ;
                          0DFE  ;       JSB     REMOVE_FROM_LIST
                          0DFE  ;
                          0DFE  ; CALLED BY:
                          0DFE  ;
                          0DFE  ;       REM_ID
                          0DFE  ;       ADD_NODE
                          0DFE  ;
                          0DFE  ; INPUTS:
                          0DFE  ;
                          0DFE  ;       ID_RESULT contains the system ID to match.
                          0DFE  ;       SYS_ID_LIST contains a list of system ID's for systems removed from
                          0DFE  ;       the display.
                          0DFE  ;       ID_COUNT contains the number of entries in the list.
                          0DFE  ;
                          0DFE  ; OUTPUTS:
                          0DFE  ;
                          0DFE  ;       Entry in SYS_ID_LIST coresponding to the the system ID specified
                          0DFE  ;       is cleared.
                          0DFE  ;       The number of entries in the list stored at ID_COUNT is decremented.
                          0DFE  ;
                          0DFE  ; RETURN STATUS:
                          0DFE  ;
                          0DFE  ;       R0 = SS$_NORMAL or failed status if entry wasn't found.
                          0DFE  ;
                          0DFE  ; REGISTER USAGE:
                          0DFE  ;
                          0DFE  ;       R4  Contains the address of the entry as returned from the
                          0DFE  ;       FIND_SYS_ID routine.
                          0DFE  ;
                          0DFE  ;--
                          0DFE
                          0DFE  REMOVE_FROM_LIST:
                  10  BB  0DFE          PUSHR   #^M<R4>
         00000E16'EF  16  0E00          JSB     FIND_SYS_ID             ; Is the entry in the list?
                0A 50 E9  0E06          BLBC    R0,99$                  ; Branch if not.
                  84  D4  0E09          CLRL    (R4)+                   ; Zero out the entry.
                  64  B4  0E0B          CLRW    (R4)                    ; All six bytes.
         000022D8'EF  D7  0E0D          DECL    ID_COUNT                ; Decrease number of entries in the
                          0E13                                          ;  list.
                  10  BA  0E13  99$:    POPR    #^M<R4>
                      05  0E15          RSB
                          0E16
                          0E16
```

```
                         0E16        .SBTTL          -       FIND_SYS_ID
                         0E16
                         0E16 ;++
                         0E16 ;
                         0E16 ; FUNCTIONAL DESCRIPTION:
                         0E16 ;
                         0E16 ;       This routine locates a system ID in the ID list.
                         0E16 ;
                         0E16 ;
                         0E16 ; CALLING SEQUENCE:
                         0E16 ;
                         0E16 ;       JSB     FIND_SYS_ID
                         0E16 ;
                         0E16 ; CALLED BY:
                         0E16 ;
                         0E16 ;       REM_ID
                         0E16 ;       CHECK_ID
                         0E16 ;
                         0E16 ; INPUTS:
                         0E16 ;
                         0E16 ;       ID_RESULT contains the system ID to match.
                         0E16 ;       SYS_ID_LIST contains a list of system ID's.
                         0E16 ;       ID_COUNT contains the number of entries in the list.
                         0E16 ;
                         0E16 ; OUTPUTS:
                         0E16 ;
                         0E16 ;       Address of entry returned in R4.
                         0E16 ;
                         0E16 ; RETURN STATUS:
                         0E16 ;
                         0E16 ;       R0 = SS$_NORMAL if the system ID is in the list.
                         0E16 ;
                         0E16 ; REGISTER USAGE:
                         0E16 ;
                         0E16 ;--
                         0E16
                         0E16 FIND_SYS_ID:
                0C   BB  0E16        PUSHR   #^M<R2,R3>
                50   D4  0E18        CLRL    R0                      ; Assume failure.
  54   00002278'EF   DE  0E1A        MOVAL   SYS_ID_LIST,R4          ; Get the address of the list.
  53   000022D8'EF   D0  0E21        MOVL    ID_COUNT,R3             ; Get the number of entries in the list.
                2F   13  0E28        BEQL    99$                     ; Branch if list is empty.
           52   02   D0  0E2A        MOVL    #2,R2                   ; Index = word.
  64   000022DC'EF   D1  0E2D 10$:   CMPL    ID_RESULT,(R4)          ; Does the entry match the designated
                         0E34                                        ;  system?
                0F   12  0E34        BNEQ    20$                     ; Branch if not.
6442   000022E0'EF   B1  0E36        CMPW    ID_RESULT+4,(R4)[R2]    ; Check all six bytes.
                05   12  0E3E        BNEQ    20$                     ; Branch if no match.
           50   01   D0  0E40        MOVL    #SS$_NORMAL,R0          ; Indicate success.
                14   11  0E43        BRB     99$                     ; Done.
                64   D5  0E45 20$:   TSTL    (R4)                    ; Is the entry empty?
                0A   12  0E47        BNEQ    30$                     ; Branch if not.
              6442   B5  0E49        TSTW    (R4)[R2]                ; Check all six bytes.
                05   12  0E4C        BNEQ    30$                     ; Branch if not empty.
           54   06   C0  0E4E        ADDL    #6,R4                   ; Point to next entry.
                DA   11  0E51        BRB     10$                     ; Loop if empty entry.
           54   06   C0  0E53 30$:   ADDL    #6,R4                   ; Point to next entry.
```

```
D4 53   F5  0E56 12489        SOBGTR  R3,10$            ; Loop for all valid entries in the
            0E59 12490                                  ;  list.
   0C   BA  0E59 12491 99$:   POPR    #^M<R2,R3>
        05  0E5B 12492        RSB
            0E5C 12493
            0E5C 12494
```

```
                              0E5C          .SBTTL          -       CONVERT_TO_HEX
                              0E5C
                              0E5C  ;++
                              0E5C  ;
                              0E5C  ; FUNCTIONAL DESCRIPTION:
                              0E5C  ;
                              0E5C  ;       This routine converts the system ID string at TKN_DSCR to a six
                              0E5C  ;       byte hex value and stores the value at ID_RESULT.
                              0E5C  ;
                              0E5C  ;
                              0E5C  ; CALLING SEQUENCE:
                              0E5C  ;
                              0E5C  ;       JSB     CONVERT_TO_HEX
                              0E5C  ;
                              0E5C  ; CALLED BY:
                              0E5C  ;
                              0E5C  ;       REM_ID
                              0E5C  ;       ADD_ID
                              0E5C  ;
                              0E5C  ; INPUTS:
                              0E5C  ;
                              0E5C  ;       TKN_DSCR contains a descriptor for the system ID string.
                              0E5C  ;
                              0E5C  ; OUTPUTS:
                              0E5C  ;
                              0E5C  ;       ID_RESULT contains the system ID.
                              0E5C  ;
                              0E5C  ; RETURN STATUS:
                              0E5C  ;
                              0E5C  ;       SS$_NORMAL or failed status returned in R0.
                              0E5C  ;
                              0E5C  ; REGISTER USAGE:
                              0E5C  ;
                              0E5C  ;--
                              0E5C
                              0E5C          .ENABL  LSB
                              0E5C
                              0E5C  CONVERT_TO_HEX:
          07FC 8F       BB    0E5C          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
               50       D4    0E60          CLRL    R0                      ; Assume failure.
52   00000129'EF        DE    0E62          MOVAL   SB$B_SYSTEMID_DSCR,R2   ; Get the address of the field
                              0E69                                          ;  descriptor for the system ID.
     02     05 A2       91    0E69          CMPB    FD$B_FORMAT(R2),#HEX    ; Is the system ID being displayed in
                              0E6D                                          ;  hexidecimal?
               24       12    0E6D          BNEQU   DECIMAL_ID              ; Branch if not.
0C   00000495'EF        B1    0E6F          CMPW    TKN_DSCR,#12            ; Are there more than 12 digits in the
                              0E76                                          ;  system ID string that was specified?
                              0E76          JGTRU   99$                     ; Error if so.
                              0E7B  ;
                              0E7B  ; Convert the string into a 6 byte hex number.
                              0E7B  ;
               06       DD    0E7B          PUSHL   #6                      ; Six byte result.
     000022DC'EF        DF    0E7D          PUSHAL  ID_RESULT               ; Address to receive result.
     00000495'EF        DF    0E83          PUSHAL  TKN_DSCR                ; String descriptor.
00000000'GF    03       FB    0E89          CALLS   #3,G^OTS$CVT_TZ_L       ; Convert the string.
             006B       31    0E90          BRW     99$                     ; Jump to common code.
                              0E93
```

```
                         0E93  DECIMAL_ID:
                         0E93
                         0E93  ;
                         0E93  ; Convert the decimal string to a six byte hex number.
                         0E93  ;
                         0E93
    52  00000495'EF  3C  0E93          MOVZWL  TKN_DSCR,R2              ; Get the number of digits in the
                         0E9A                                           ;  string.
    55  00000499'EF  D0  0E9A          MOVL    TKN_DSCR+4,R5            ; Get the address of the string.
              14  52  D1  0EA1          CMPL    R2,#20                   ; Are there more that twenty digits in
                         0EA4                                           ;  the string?
                         0EA4          JGTRU   99$                      ; Error if so.
        53    52  03  9C  0EA9          ROTL    #3,R2,R3                 ; Multiply number of digits by 8 to get
                         0EAD                                           ;  quadword index.
    54  0000056A'EF  DE  0EAD          MOVAL   DEC_TAB_END,R4           ; Get the address of the end of the
                         0EB4                                           ;  decimal conversion table.
              54  53  C2  0EB4          SUBL    R3,R4                    ; Get the address of the first quadword
                         0EB7                                           ;  in the table to use for conversion.
        000022DC'EF  D4  0EB7          CLRL    ID_RESULT                ; Clear the result of conversion.
        000022E0'EF  D4  0EBD          CLRL    ID_RESULT+4              ;
              56  84  7D  0EC3  10$:    MOVQ    (R4)+,R6                 ; Get the quadword.
        53    65  30  83  0EC6          SUBB3   #^X30,(R5),R3            ; Convert the ascii digit to decimal.
                  2A  13  0ECA          BEQLU   40$                      ; Ignore if digit is zero.
                         0ECC          JLSSU   99$                      ; Branch if not valid decimal number.
              09  53  91  0ED1          CMPB    R3,#9                    ; Check high limit.
                  28  1A  0ED4          BGTRU   99$                      ; Branch if not valid decimal number.
                         0ED6  ;
                         0ED6  ; Multiply the quadword in R6,R7 by the byte in R3 and add the result to
                         0ED6  ; ID_RESULT.
                         0ED6  ;
              53  53  9A  0ED6          MOVZBL  R3,R3                    ; Convert byte to longword.
    58    00  56  53  7A  0ED9          EMUL    R3,R6,#0,R8              ; Multiply low longword.
                  03  18  0EDE          BGEQ    20$                      ; Branch if result is positive.
              59  53  C0  0EE0          ADDL    R3,R9                    ; Otherwise, compensate for unsighned
                         0EE3                                           ;  bias.
    59    59  57  53  7A  0EE3  20$:    EMUL    R3,R7,R9,R9              ; Multiply high longword.
   000022DC'EF    58  C0  0EE8  30$:    ADDL    R8,ID_RESULT             ; Add the resulting quadword to
   000022E0'EF    59  D8  0EEF          ADWC    R9,ID_RESULT+4           ;  ID_RESULT.
                  55  D6  0EF6  40$:    INCL    R5                       ; Point to next byte of string.
              C8  52  F5  0EF8          SOBGTR  R2,10$                   ; Loop for all digits in the string.
              50  01  D0  0EFB          MOVL    #SS$_NORMAL,R0           ; Indicate success.
                         0EFE
         07FC  8F  BA  0EFE  99$:    POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
                  05  0F02          RSB
                         0F03
                         0F03          .DSABL  LSB
                         0F03
                         0F03
```

```
                              0F03        .SBTTL          -       SET_WIDTH
                              0F03
                              0F03  ;++
                              0F03  ;
                              0F03  ; FUNCTIONAL DESCRIPTION:
                              0F03  ;
                              0F03  ;       This routine is used to change the number of columns used by a
                              0F03  ;       particular field.  It changes the value in the field descriptor at
                              0F03  ;       location FD$B_FIELD_SIZE after checking that the desired width does not
                              0F03  ;       exceed the minimum or maximum values for the field.
                              0F03  ;
                              0F03  ; CALLING SEQUENCE:
                              0F03  ;
                              0F03  ;       CALLS   #0,SET_WIDTH
                              0F03  ;
                              0F03  ; CALLED BY:
                              0F03  ;
                              0F03  ;       READ_W_PROMPT
                              0F03  ;
                              0F03  ; INPUTS:
                              0F03  ;
                              0F03  ;       Desired width stored at location VALUE.
                              0F03  ;       Address of the field descriptor for the selected field stored in
                              0F03  ;       KEY_DSCR.
                              0F03  ;
                              0F03  ; OUTPUTS:
                              0F03  ;
                              0F03  ;       Value at location VALUE stored at offset FD$B_FIELD_SIZE in the field
                              0F03  ;       descriptor for the selected field.
                              0F03  ;
                              0F03  ; RETURN STATUS:
                              0F03  ;
                              0F03  ;       SS$_NORMAL or failed status returned in R0.
                              0F03  ;
                              0F03  ; REGISTER USAGE:
                              0F03  ;
                              0F03  ;       R4 -    Points to the base address of the field descriptor for the
                              0F03  ;               selected field.
                              0F03  ;--
                              0F03
                        0010  0F03  .ENTRY  SET_WIDTH,^M<R4>
54      0000054B'EF     D0    0F05          MOVL    KEY_DSCR,R4             ; Get the field descriptor address.
        0000054F'EF     91    0F0C          CMPB    VALUE,-                 ; Is the desired size smaller than the
                01 A4         0F12                  FD$B_MIN_FLD_SIZE(R4)   ;  minimum allowed for this field?
                   13   1F    0F14          BLSSU   10$                     ; Branch if so.
        0000054F'EF     91    0F16          CMPB    VALUE,-                 ; Is the desired size larger than the
                03 A4         0F1C                  FD$B_MAX_FLD_SIZE(R4)   ;  maximum allowed for this field?
                   0F   1A    0F1E          BGTRU   20$                     ; Branch if so.
        0000054F'EF     90    0F20          MOVB    VALUE,-                 ; Store the new field size.
                   64         0F26                  FD$B_FIELD_SIZE(R4)     ;
                   0A   11    0F27          BRB     99$                     ; Jump to common code.
                01 A4   90    0F29  10$:    MOVB    FD$B_MIN_FLD_SIZE(R4),- ; Store the minimum field size.
                   64         0F2C                  FD$B_FIELD_SIZE(R4)
                   04   11    0F2D          BRB     99$                     ; Jump to common code.
                03 A4   90    0F2F  20$:    MOVB    FD$B_MAX_FLD_SIZE(R4),- ; Store the maximum field size.
                   64         0F32                  FD$B_FIELD_SIZE(R4)     ;
                        04    0F33  99$:    RET
```

```
0F34 12659
0F34 12660
```

```
0F34        .SBTTL      -       SET_FORMAT
0F34  ;++
0F34  ;
0F34  ; FUNCTIONAL DESCRIPTION:
0F34  ;
0F34  ;       This routine is used to change the output format of a particular field
0F34  ;       to decimal or hexadecimal.  It changes the value in the field
0F34  ;       descriptor at location FD$B_FORMAT after checking that the current
0F34  ;       format is not ascii or ascic.  FD$B_MAX_DATA_SIZE is set to the maximum
0F34  ;       number of columns needed to display the data in the new format.  This
0F34  ;       value is also compared with FD$B_TITLE_SIZE (the number of characters
0F34  ;       in the field title) and the larger of the two is saved as the maximum
0F34  ;       number of columns necessary to display the field (FD$B_MAX_FLD_SIZE).
0F34  ;
0F34  ;       Note:   This routine will return an error if the number of bytes of
0F34  ;               data needed to display the field exceeds eight and the format
0F34  ;               is being changed from hexadecimal to decimal.  This is because
0F34  ;               the format routines will not format a value larger than a
0F34  ;               quadword in decimal.
0F34  ;
0F34  ; CALLING SEQUENCE:
0F34  ;
0F34  ;       CALLS   #0,SET_FORMAT
0F34  ;
0F34  ; CALLED BY:
0F34  ;
0F34  ;       READ_W_PROMPT
0F34  ;
0F34  ; INPUTS:
0F34  ;
0F34  ;       Desired format type stored at location TPARSE_BLK+TPA$L_PARAM.
0F34  ;       Address of the field descriptor for the selected field stored in
0F34  ;       KEY_DSCR.
0F34  ;
0F34  ; OUTPUTS:
0F34  ;
0F34  ;       Value at location TPARSE_BLK+TPA$L_PARAM stored at offset
0F34  ;       FD$B_FORMAT in the field descriptor for the selected field.
0F34  ;
0F34  ;       Value at offset FD$B_MAX_DATA_SIZE in the field descriptor set to the
0F34  ;       maximum number of columns needed to display the data.
0F34  ;
0F34  ;       Value at offset FD$B_MAX_FLD_SIZE in the field descriptor set to the
0F34  ;       maximum number of columns needed to display the field.
0F34  ;
0F34  ; RETURN STATUS:
0F34  ;
0F34  ;       SS$_NORMAL or failed status returned in R0.
0F34  ;
0F34  ; REGISTER USAGE:
0F34  ;
0F34  ;       R3 -    Used to hold the number of byte of SCS data needed to display
0F34  ;               the field.
0F34  ;       R4 -    Points to the base address of the field descriptor for the
0F34  ;               selected field.
0F34  ;       R5 -    Pointer to data length to number of columns conversion table.
0F34  ;--
```

```
                              0F34
                        0038  0F34       .ENTRY  SET_FORMAT,^M<R3,R4,R5>
 54       0000054B'EF     D0  0F36               MOVL    KEY_DSCR,R4                ; Get the field descriptor address.
          000004E7'EF     91  0F3D               CMPB    TPARSE_BLK+TPA$L_PARAM,-; Is the format already set to the
                   05 A4      0F43                       FD$B_FORMAT(R4)            ;  desired type?
                      50  13  0F45               BEQL    99$                        ; Return if so.
         01        05 A4  91  0F47               CMPB    FD$B_FORMAT(R4),#ASCII     ; Is the field formated in ascii?
                      48  13  0F4B               BEQL    98$                        ; Error if so.
         04        05 A4  91  0F4D               CMPB    FD$B_FORMAT(R4),#ASCIC     ; Is the field formated in ascii using
                              0F51                                                  ;  a counted ascii string?
                      42  13  0F51               BEQL    98$                        ; Error if so.
         53        04 A4  9A  0F53               MOVZBL  FD$B_LENGTH(R4),R3         ; Get the number of bytes of SCS data
                              0F57                                                  ;  needed to display this field.
          000004E7'EF     91  0F57               CMPB    TPARSE_BLK+TPA$L_PARAM,-; Is the new format decimal?
                      03      0F5D                       #DEC                       ;
                      13  12  0F5E               BNEQ    10$                        ; Branch if not.
              08      53  91  0F60               CMPB    R3,#8                      ; Is the SCS data more that 8 bytes?
                      30  14  0F63               BGTR    98$                        ; Error if so.
 55       000004A9'EF     DE  0F65               MOVAL   CVT_TABLE,R5               ; Point to conversion table.
       02 A4     6543     90  0F6C               MOVB    (R5)[R3],-                 ; Save the maximum number of columns
                              0F71                       FD$B_MAX_DATA_SIZE(R4)     ;  needed to display the field in
                              0F71                                                  ;  decimal.
                      05  11  0F71               BRB     20$                        ; Jump to common code.
    02 A4     53      53  81  0F73       10$:    ADDB3   R3,R3,-                    ; Save the maximum number of columns
                              0F78                       FD$B_MAX_DATA_SIZE(R4)     ;  needed to display the field in
                              0F78                                                  ;  hexadecimal.
          000004E7'EF     90  0F78       20$:    MOVB    TPARSE_BLK+TPA$L_PARAM,-; Set the new format type.
                   05 A4      0F7E                       FD$B_FORMAT(R4)
                   02 A4  91  0F80               CMPB    FD$B_MAX_DATA_SIZE(R4),-; Are more columns required to display
                   18 A4      0F83                       FD$B_TITLE_SIZE(R4)        ;  the data than to display the field
                              0F85                                                  ;  title?
                      07  1F  0F85               BLSSU   30$                        ; Branch if not.
                   02 A4  90  0F87               MOVB    FD$B_MAX_DATA_SIZE(R4),-; Make the maximum field size the same
                   03 A4      0F8A                       FD$B_MAX_FLD_SIZE(R4)      ;  as the maximum data size.
                      09  11  0F8C               BRB     99$                        ; Return with success.
                   18 A4  90  0F8E       30$:    MOVB    FD$B_TITLE_SIZE(R4),-      ; Make the maximum field size the same
                   03 A4      0F91                       FD$B_MAX_FLD_SIZE(R4)      ;  as the title size.
                      02  11  0F93               BRB     99$                        ; Return with success.
                      50  D4  0F95       98$:    CLRL    R0                         ; Indicate failure.
                          04  0F97       99$:    RET
                              0F98
                              0F98
```

```
                                  0F98 12762          .SBTTL          -        SET_INTERVAL
                                  0F98 12763 ;++
                                  0F98 12764 ;
                                  0F98 12765 ; FUNCTIONAL DESCRIPTION:
                                  0F98 12766 ;
                                  0F98 12767 ;        This routine converts the number of seconds string entered in the SET
                                  0F98 12768 ;        INTERVAL command to binary time and stores it at location DELAY.
                                  0F98 12769 ;
                                  0F98 12770 ; CALLING SEQUENCE:
                                  0F98 12771 ;
                                  0F98 12772 ;        CALLS   #0,SET_INTERVAL
                                  0F98 12773 ;
                                  0F98 12774 ; CALLED BY:
                                  0F98 12775 ;
                                  0F98 12776 ;        READ_W_PROMPT
                                  0F98 12777 ;
                                  0F98 12778 ; INPUTS:
                                  0F98 12779 ;
                                  0F98 12780 ;        Desired number of seconds stored at location VALUE.
                                  0F98 12781 ;
                                  0F98 12782 ; OUTPUTS:
                                  0F98 12783 ;
                                  0F98 12784 ;        Binary representation of desired interval stored at location DELAY.
                                  0F98 12785 ;
                                  0F98 12786 ; RETURN STATUS:
                                  0F98 12787 ;
                                  0F98 12788 ;        SS$_NORMAL returned in R0.
                                  0F98 12789 ;
                                  0F98 12790 ;--
                                  0F98 12791
                             0000 0F98 12792 .ENTRY  SET_INTERVAL,^M<>
                                  0F9A 12793
   0000054F'EF  0000054F'EF   3C  0F9A 12794         MOVZWL  VALUE,VALUE              ; Make sure VALUE is a longword.
00 0000054F'EF  FF676980 8F   7A  0FA5 12795         EMUL    #-10000000,VALUE,#0,-    ; Convert the number of seconds to
                000021AC'EF       0FB1 12796                 DELAY                    ;  binary time.
                      50   01 D0  0FB6 12797         MOVL    #SS$_NORMAL,R0           ; Return success.
                              04  0FB9 12798         RET                              ; Return with status.
                                  0FBA 12799
                                  0FBA 12800
```

```
                              0FBA         .SBTTL          -       SET_SCREEN
                              0FBA       ;++
                              0FBA       ;
                              0FBA       ; FUNCTIONAL DESCRIPTION:
                              0FBA       ;
                              0FBA       ;       This routine sets the width of the terminal screen to 80 or 132
                              0FBA       ;       columns.  If the desired width is 80 or less than the screen is set to
                              0FBA       ;       80 columns.  If the desired width is 81 or greater than the screen is
                              0FBA       ;       set to 132 columns.
                              0FBA       ;       If this routine is not being called while initialization or exit is
                              0FBA       ;       in progress and is being run on a video terminal than the display
                              0FBA       ;       header is redrawn.
                              0FBA       ;
                              0FBA       ; CALLING SEQUENCE:
                              0FBA       ;
                              0FBA       ;       CALLS   #0,SET_SCREEN
                              0FBA       ;
                              0FBA       ; CALLED BY:
                              0FBA       ;
                              0FBA       ;       INIT
                              0FBA       ;       MAIN_ERR
                              0FBA       ;       READ_W_PROMPT
                              0FBA       ;
                              0FBA       ; INPUTS:
                              0FBA       ;
                              0FBA       ;       Desired screen width stored at location VALUE.
                              0FBA       ;
                              0FBA       ; OUTPUTS:
                              0FBA       ;
                              0FBA       ;       SCREEN_WIDTH is set to 80 or 132.
                              0FBA       ;
                              0FBA       ; IMPLICIT OUTPUTS:
                              0FBA       ;
                              0FBA       ;       Terminal is set to 80 or 132 column mode.
                              0FBA       ;
                              0FBA       ; RETURN STATUS:
                              0FBA       ;
                              0FBA       ;       SS$_NORMAL or failed status returned in R0.
                              0FBA       ;
                              0FBA       ;--
                              0FBA
                        0000  0FBA       .ENTRY  SET_SCREEN,^M<>
                              0FBC
             50    01   D0    0FBC               MOVL    #SS$_NORMAL,R0                  ; Assume success.
 0050 8F   0000054F'EF  B1    0FBF               CMPW    VALUE,#NARROW                   ; Is the desired screen width > 80 ?
                   44   1A    0FC8               BGTRU   10$                             ; Branch if so.
00002222'EF   0050 8F   B1    0FCA               CMPW    #NARROW,SCREEN_WIDTH            ; Is the terminal width already 80?
                              0FD3               JEQL    99$                             ; Return if it is.
00002222'EF   0050 8F   B0    0FD8               MOVW    #NARROW,SCREEN_WIDTH            ; Set number of columns to 80.
  76 00002247'EF   01   E1    0FE1               BBC     #DVT$V_DCRT,OUT_DEV,30$         ; Branch if terminal isn't a DECcrt.
        0000014E'EF     DF    0FE9               PUSHAL  DCRT_NARROW_DSCR                ; Push address of descriptor to set
                              0FEF                                                       ;  terminal into 80 column mode.
  5E 00002247'EF   04   E0    0FEF               BBS     #DVT$V_AVO,OUT_DEV,20$          ; Branch if terminal is a DECcrt with
                              0FF7                                                       ;  advanced video option.
     00002260'EF   18   D0    0FF7               MOVL    #24,LAST_LINE                   ; Save the number of the last line
                              0FFE                                                       ;  (bottom command line) on the
                              0FFE                                                       ;  terminal.
```

```
         0000225C'EF   17  D0  0FFE 12859        MOVL    #23,CMD_LINE              ; Save the number of the next-to-last
                               1005 12860                                          ;  line (top command line) on the
                               1005 12861                                          ;  terminal.
         00002258'EF   16  D0  1005 12862        MOVL    #22,LAST_DATA_LINE        ; Save the number of the last line on
                               100C 12863                                          ;  the terminal to be used to display
                               100C 12864                                          ;  data.
                       47  11  100C 12865        BRB     20$                       ; Jump to common code.
      60 00002247'EF   00  E0  100E 12866 10$:   BBS     #DVT$V_VT52,OUT_DEV,99$   ; Branch if terminal is VT52 type.
 00002222'EF    0084   8F  B1  1016 12867        CMPW    #WIDE,SCREEN_WIDTH        ; Is the terminal width already 132?
                       55  13  101F 12868        BEQL    99$                       ; Return if it is.
 00002222'EF    0084   8F  B0  1021 12869        MOVW    #WIDE,SCREEN_WIDTH        ; Set number of columns to 132.
      44 00002247'EF   01  E1  102A 12870        BBC     #DVT$V_DCRT,OUT_DEV,99$   ; Branch if terminal isn't a DECcrt.
      15 00002247'EF   04  E0  1032 12871        BBS     #DVT$V_AVO,OUT_DEV,15$    ; Branch if DECcrt with advanced video
                               103A 12872                                          ;  option.
         00002260'EF   0E  D0  103A 12873        MOVL    #14,LAST_LINE             ; Save the number of the last line
                               1041 12874                                          ;  (bottom command line) on the
                               1041 12875                                          ;  terminal.
         0000225C'EF   0D  D0  1041 12876        MOVL    #13,CMD_LINE              ; Save the number of the next-to-last
                               1048 12877                                          ;  line (top command line) on the
                               1048 12878                                          ;  terminal.
         00002258'EF   0C  D0  1048 12879        MOVL    #12,LAST_DATA_LINE        ; Save the number of the last line on
                               104F 12880                                          ;  the terminal to be used to display
                               104F 12881                                          ;  data.
            0000015B'EF    DF  104F 12882 15$:   PUSHAL  DCRT_WIDE_DSCR            ; Push address of descriptor to set
                               1055 12883                                          ;  terminal into 132 column mode.
         00000000'GF   01  FB  1055 12884 20$:   CALLS   #1,G^LIB$PUT_OUTPUT       ; Set the terminal.
                    17 50  E9  105C 12885        BLBC    R0,99$                    ; Branch on error.
                       01  E0  105F 12886 30$:   BBS     #MF$V_EXIT_FLAG,-         ; Branch if exit flag is set.
      0F 00000476'EF           1061 12887                MISC_FLAGS,99$            ;
                       00  E0  1067 12888        BBS     #MF$V_INIT_FLAG,-         ; Or if INIT flag is set.
      07 00000476'EF           1069 12889                MISC_FLAGS,99$            ;
         00002348'EF   00  FB  106F 12890        CALLS   #0,REFRESH                ; Refresh the screen.
                           04  1076 12891 99$:   RET
                               1077 12892
                               1077 12893
```

```
                              1077 12895         .SBTTL          -       GET_HELP
                              1077 12896 ;++
                              1077 12897 ;
                              1077 12898 ; FUNCTIONAL DESCRIPTION:
                              1077 12899 ;
                              1077 12900 ;       This routine is called when the user asks for help.  The remainder
                              1077 12901 ;       of the string is passed to the LBR$OUTPUT_HELP routine.
                              1077 12902 ;
                              1077 12903 ; CALLING SEQUENCE:
                              1077 12904 ;
                              1077 12905 ;       CALLS   #0,GET_HELP
                              1077 12906 ;
                              1077 12907 ; CALLED BY:
                              1077 12908 ;
                              1077 12909 ;       READ_W_PROMPT
                              1077 12910 ;
                              1077 12911 ; IMPLICIT_INPUTS:
                              1077 12912 ;
                              1077 12913 ;       TPARSE_BLK+TPA$L_STRINGPTR points to the first keyword in the command
                              1077 12914 ;       buffer following the ''HELP'' command string.
                              1077 12915 ;
                              1077 12916 ;       CMD_LINE_DSCR contains the number of bytes in the entire command line.
                              1077 12917 ;
                              1077 12918 ; IMPLICIT OUTPUTS:
                              1077 12919 ;
                              1077 12920 ;       Help txt is output to the terminal.
                              1077 12921 ;
                              1077 12922 ; RETURN STATUS:
                              1077 12923 ;
                              1077 12924 ;       SS$_NORMAL or failed status returned in R0.
                              1077 12925 ;
                              1077 12926 ;--
                              1077 12927
                        0800  1077 12928 .ENTRY  GET_HELP,^M<R11>
                              1079 12929
                              1079 12930 ;
                              1079 12931 ; Store the number of bytes remaining in the command string in HELP_DSCR.
                              1079 12932 ;
        000020D7'EF     C3    1079 12933         SUBL3   CMD_LINE_DSCR+4,-               ; Get the number of bytes we've
        000004D3'EF           107F 12934                 TPARSE_BLK+TPA$L_STRINGPTR,-;  already parsed.
                 50           1084 12935                 R0
                              1085 12936
     000020D3'EF   50   C3    1085 12937         SUBL3   R0,CMD_LINE_DSCR,-      ; Save the number of bytes remaining.
        00000553'EF           108C 12938                 HELP_DSCR               ;
                              1091 12939 ;
                              1091 12940 ; Store the address of remainder of the command string in HELP_DSCR+4.
                              1091 12941 ;
        000004DB'EF     D0    1091 12942         MOVL    TPARSE_BLK+TPA$L_TOKENPTR,-
        00000557'EF           1097 12943                 HELP_DSCR+4
                              109C 12944
     00002248'EF   06   D0    109C 12945         MOVL    #HEADER_END-1,LINE_NUM  ; Set the scrolling region to the area
             104B'      30    10A3 12946         BSBW    DRAW_LAST_LINE          ; Draw bottom line of the display box
                              10A6 12947                                         ;  as the bottom line of the display
                              10A6 12948                                         ;  header.
                              10A6 12949         JLBC    R0,99$                  ; Branch on error.
  09 00002247'EF   01   E1    10AC 12950         BBC     #DVT$V_DCRT,OUT_DEV,10$ ; Branch if terminal is not a DECcrt.
        00001A03'EF     16    10B4 12951         JSB     SET_REGION_HELP         ; Establish a scrolling region.
```

```
               6B 50   E9  10BA             BLBC    R0,99$                  ; Branch on error.
     00002248'EF   07   D0  10BD      10$:   MOVL    #HEADER_END,LINE_NUM    ; Set cursor to line after header.
     0000224C'EF        3F  10C4             PUSHAW  COL_NUM                 ; Erase all the data.
     00002248'EF        3F  10CA             PUSHAW  LINE_NUM                ;
     00000000'GF   02   FB  10D0             CALLS   #2,G^LIB$ERASE_PAGE     ;
               4E 50   E9  10D7             BLBC    R0,99$                  ; Branch on error.
     00000000'GF        DF  10DA             PUSHAL  G^LIB$GET_INPUT         ; Push address of input routine.
     0000047D'EF        DF  10E0             PUSHAL  HELP_FLAGS              ; Push address of flags.
     0000010C'EF        DF  10E6             PUSHAL  LIBRARY_NAME            ; Push address of descriptor for the
                            10EC                                             ;  library name.
     00000553'EF        DF  10EC             PUSHAL  HELP_DSCR               ; Push address of descriptor for the
                            10F2                                             ;  help keywords.
     00002222'EF        3C  10F2             MOVZWL  SCREEN_WIDTH,-          ; Save width of screen.
     0000222C'EF            10F8                     OUTPUT_WIDTH            ;
     0000222C'EF        DF  10FD             PUSHAL  OUTPUT_WIDTH            ; Push address of longword containing
                            1103                                             ;  the output width.
     00000000'GF        DF  1103             PUSHAL  G^LIB$PUT_OUTPUT        ; Push address of output routine.
     00000000'GF   06   FB  1109             CALLS   #6,G^LBR$OUTPUT_HELP    ; Call librarian help routine.
               15 50   E9  1110             BLBC    R0,99$                  ; Branch on error.
  03 00002247'EF   01   E1  1113             BBC     #DVT$V_DCRT,OUT_DEV,20$ ; Branch if terminal is not a DECcrt.
            08B0'       30  111B             BSBW    SET_REGION_FULL         ; Set scrolling region to use full
                            111E                                             ;  screen.
     00002348'EF   00   FB  111E      20$:   CALLS   #0,REFRESH              ; Refresh the screen.
               50  01   D0  1125             MOVL    #SS$_NORMAL,R0
                        04  1128      99$:   RET
                            1129
                            1129
```

```
                                  1129  .SBTTL          -       RESET
                                  1129 ;++
                                  1129 ;
                                  1129 ; FUNCTIONAL DESCRIPTION:
                                  1129 ;
                                  1129 ;       This routine restores the default classes and fields for the cluster
                                  1129 ;       and local reports.  It also reselects the cluster report, reenables
                                  1129 ;       any systems that have been removed from the report, and resets the
                                  1129 ;       screen to 80 columns if run on a DECcrt.
                                  1129 ;
                                  1129 ; CALLING SEQUENCE:
                                  1129 ;
                                  1129 ;       CALLS   #0,RESET
                                  1129 ;
                                  1129 ; CALLED BY:
                                  1129 ;
                                  1129 ;       READ_W_PROMPT
                                  1129 ;
                                  1129 ; OUTPUTS
                                  1129 ;
                                  1129 ;       Cluster bitmap restored using copy saved by the INIT routine.
                                  1129 ;       Local bitmap restored using copy saved by the INIT routine.
                                  1129 ;       CURRENT_REPORT set to cluster.
                                  1129 ;       Default cluster report classes stored in CLASS_SELECT.
                                  1129 ;       Default local report classes stored in LOCAL_SAVE.
                                  1129 ;
                                  1129 ; RETURN STATUS:
                                  1129 ;
                                  1129 ;       None.
                                  1129 ;
                                  1129 ;--
                                  1129
                           00FC   1129 .ENTRY  RESET,^M<R2,R3,R4,R5,R6,R7>
                                  112B
0000054F'EF   00000050 8F   D0    112B         MOVL    #80,VALUE                    ; Screen width = 80 columns.
                                  1136
                                  1136 ;
                                  1136 ; Set default classes for each report.
                                  1136 ;
              00370006 8F   D0    1136         MOVL    #CLS$M_ALL_SYS!CLS$M_CSB,-
              00002204'EF         113C                 CLUST_SAVE
     EEB7 CF  00000180 8F   D0    1141         MOVL    #CLS$M_LPRT!CLS$M_ERR,CLASS_SELECT
                                  114A ;
                                  114A ; Restore saved copy of cluster bitmap.
                                  114A ;
                    55 02   D0    114A         MOVL    #CLUSTER_MAP_LEN,R5          ; Get the length of the cluster bitmap.
          56  00000000'EF   DE    114D         MOVAL   CLUSTER_BITMAP,R6            ; Get the address of the cluster
                                  1154                                              ;  bitmap.
          57  0000000C'EF   DE    1154         MOVAL   CL_SAVE_BITMAP,R7            ; Get the address of the saved copy of
                                  115B                                              ;  the cluster bitmap.
                    86 87   D0    115B 60$:    MOVL    (R7)+,(R6)+                  ; Restore the cluster bitmap.
                    FA 55   F4    115E         SOBGEQ  R5,60$                       ;
                                  1161
                                  1161 ;
                                  1161 ; Restore the field sizes for the cluster report.
                                  1161 ;
                                  1161
```

```
        57  00000000'EF  DE  1161        MOVAL   CLUSTER_SIZE_ARRAY,R7      ; Get the address of the cluster field
                             1168                                           ;  size array.
        56  00000000'EF  DE  1168        MOVAL   CLUSTER_TABLE,R6           ; Get the address of the cluster
                             116F                                           ;   address table.
        55  00000046 8F  D0  116F        MOVL    #NUM_CLUSTER_DSCRS,R5      ; Get the number of field descriptors
                             1176                                           ;  for the cluster report.
                     53  D4  1176        CLRL    R3                         ; Clear index.
             52    6643  D0  1178  70$:  MOVL    (R6)[R3],R2                ; Get the address of the field
                             117C                                           ;  descriptor.
             62    6743  90  117C        MOVB    (R7)[R3],-                 ; Copy the original field size back
                             1180                FD$B_FIELD_SIZE(R2)        ;  into the field descriptor.
          F4 53      55  F2  1180        AOBLSS  R5,R3,70$                  ; Loop for each field descriptor in the
                             1184                                           ;  cluster report.
                             1184  ;
                             1184  ; Restore saved copy of the local ports bitmap.
                             1184  ;
             55      00  D0  1184        MOVL    #LOCAL_MAP_LEN,R5          ; Get the length of the local bitmap.
        56  00000000'EF  DE  1187        MOVAL   LOCAL_BITMAP,R6            ; Get the address of the local bitmap.
        57  00000004'EF  DE  118E        MOVAL   LOC_SAVE_BITMAP,R7         ; Get the address of the saved copy of
                             1195                                           ;  the local bitmap.
             86      87  D0  1195  80$:  MOVL    (R7)+,(R6)+                ; Restore the local ports bitmap.
                  FA 55  F4  1198        SOBGEQ  R5,80$                     ;
                             119B
                             119B  ;
                             119B  ; Restore the field sizes for the local report.
                             119B  ;
                             119B
        57  00000000'EF  DE  119B        MOVAL   LOCAL_SIZE_ARRAY,R7        ; Get the address of the local field
                             11A2                                           ;  size array.
        56  00000000'EF  DE  11A2        MOVAL   LOCAL_TABLE,R6             ; Get the address of the local
                             11A9                                           ;   address table.
             55      19  D0  11A9        MOVL    #NUM_LOCAL_DSCRS,R5        ; Get the number of field descriptors
                             11AC                                           ;  for the local report.
                     53  D4  11AC        CLRL    R3                         ; Clear index.
             52    6643  D0  11AE  90$:  MOVL    (R6)[R3],R2                ; Get the address of the field
                             11B2                                           ;  descriptor.
             62    6743  90  11B2        MOVB    (R7)[R3],-                 ; Copy the original field size back
                             11B6                FD$B_FIELD_SIZE(R2)        ;  into the field descriptor.
          F4 53      55  F2  11B6        AOBLSS  R5,R3,90$                  ; Loop for each field descriptor in the
                             11BA                                           ;  local report.
                             11BA
00   00002278'EF     00  2C  11BA        MOVC5   #0,SYS_ID_LIST,#0,-        ; Restore any systems that have been
00002278'EF     0060 8F      11C2                #ID_LIST_LEN,SYS_ID_LIST;  ;  removed from the display.
            000022D8'EF  D4  11CA        CLRL    ID_COUNT                   ; Clear the number of system removed
                             11D0                                           ;  from the display.
                             11D0
            0000047A'EF  94  11D0        CLRB    SCROLL_FLAGS               ; Clear the scroll flags.
            0000047C'EF  94  11D6        CLRB    CLSTR_SCROLL_FLAGS         ; Clear the flags that will be retored
                             11DC                                           ;  as the scroll flags after call to
                             11DC                                           ;  SHOW_CLUSTER.
            00000464'EF  D4  11DC        CLRL    CLSTR_H_SCROLL             ; Clear all the scroll counters.
            00000468'EF  D4  11E2        CLRL    LOCAL_H_SCROLL             ;
            0000045C'EF  D4  11E8        CLRL    V_SCROLL                   ;
                     01  8A  11EE        BICB    #RPT$M_CLSTR,-             ; Force the SHOW_CLUSTER routine to
                EE0D CF      11F0                CURRENT_REPORT             ;  reset the report by setting report
                             11F3                                           ;  to the local report.
        52  00000476'EF  D0  11F3        MOVL    MISC_FLAGS,R2              ; Save copy of miscellaneous flags.
```

```
              01  C8  11FA 13094          BISL    #MF$M_INIT_FLAG,-       ; Set init flag so we don't draw the
    00000476'EF      11FC 13095                  MISC_FLAGS              ;  local report.
  FDB4 CF     00  FB  1201 13096          CALLS   #0,SET_SCREEN           ; Set the screen width.
                      1206 13097
                      1206 13098 ASSUME   MF$V_INIT_FLAG EQ 0
                      1206 13099
  01     00   52  F0  1206 13100          INSV    R2,#MF$V_INIT_FLAG,#1,- ; Restore init flag to original state.
    00000476'EF      120A 13101                  MISC_FLAGS              ;
  F715 CF     00  FB  120F 13102          CALLS   #0,SHOW_CLUSTER         ; Set default report to clusters.
                  04  1214 13103          RET
                      1215 13104
                      1215 13105
```

```
                                  1215         .SBTTL   -        EXECUTE
                                  1215  ;++
                                  1215  ;
                                  1215  ; FUNCTIONAL DESCRIPTION:
                                  1215  ;
                                  1215  ;         This routine is used to execute a command procedure when requested by
                                  1215  ;         the user (@filespec), to execute a startup initialization file
                                  1215  ;         when Show Cluster if first envoked, and to read commands when the input
                                  1215  ;         is from a file.
                                  1215  ;
                                  1215  ; CALLING SEQUENCE:
                                  1215  ;
                                  1215  ;         CALLS   #0,EXECUTE
                                  1215  ;
                                  1215  ; CALLED BY:
                                  1215  ;
                                  1215  ;         INIT                                  ; Execute initialization file.
                                  1215  ;         READ_W_PROMPT                         ; Execute command procedure.
                                  1215  ;         END_REPORT                            ; Get command input from a file.
                                  1215  ;
                                  1215  ; RETURN STATUS:
                                  1215  ;
                                  1215  ;         SS$_NORMAL or falied status returned in R0.
                                  1215  ;
                                  1215  ;--
                                  1215
                           003C   1215  .ENTRY   EXECUTE,^M<R2,R3,R4,R5>
                                  1217
  20   000000CC'EF   00     2C    1217           MOVC5   #0,IN_FILE_SPEC,#^A/ /,-; Pad the filespec with blanks.
              0050 8F             121F                   #IN_FILE_SPEC_LEN,-
         000000CC'EF              1222                   IN_FILE_SPEC          ;
                                  1227
                  00        E1    1227  3$:      BBC     #MF$V_INIT_FLAG,-     ; Branch if initialization not in
         12 00000476'EF           1229                   MISC_FLAGS,4$         ;  progress.
0000012E'FF   0000012A'EF   28    122F           MOVC3   INIT_NAME,@INIT_NAME+4,-; Store logical name in the file spec.
         000000CC'EF              123A                   IN_FILE_SPEC          ;
                  2A        11    123F           BRB     7$                    ; Jump to common.
                                  1241
   12 00002246'EF   05      E1    1241  4$:      BBC     #DVT$V_DISK,IN_DEV,5$ ; Branch if input isn't from a file.
         000000DE'EF        28    1249           MOVC3   IN_DEV_NAME,-         ; Otherwise SYS$INPUT is the filespec.
         000000E6'EF              124F                   IN_DEV_NAME+8,-       ;
         000000CC'EF              1254                   IN_FILE_SPEC          ;
                  10        11    1259           BRB     7$                    ; Jump to common.
                                  125B
                                  125B  ;
                                  125B  ; Get the filename from the command line and store in in the filespec.
                                  125B  ;
                                  125B
         000004CF'EF        28    125B  5$:      MOVC3   TPARSE_BLK+TPA$L_STRINGCNT,-
         000004D3'FF              1261                   @TPARSE_BLK+TPA$L_STRINGPTR,-
         000000CC'EF              1266                   IN_FILE_SPEC
                                  126B
                                  126B  7$:      $PARSE  FAB = INFAB           ; Expand the filespec.
                                  1278           JLBC    R0,20$                ; Branch on error.
                                  127E
                                  127E           $OPEN   FAB = INFAB           ; Open the file.
                                  128B           JLBC    R0,20$                ; Branch on error.
```

```
00000476'EF   00000040 8F   C8   1291 13164         BISL    #MF$M_IN_OPEN,MISC_FLAGS; Indicate file is open.
                                 129C 13165
                                 129C 13166         $CONNECT -                      ; Connect to input.
                                 129C 13167                 RAB = INRAB             ;
                   60 50    E9   12A9 13168         BLBC    R0,20$                  ; Branch on error.
                                 12AC 13169
                                 12AC 13170 10$:    $GET    RAB = INRAB             ; Get a record.
                   50 50    E9   12B9 13171         BLBC    R0,20$                  ; Branch on error.
                                 12BC 13172
00000310'EF   000002EE'EF   B0   12BC 13173         MOVW    INRAB+RAB$W_RSZ,REC_DSCR; Get the record size.
              00000310'EF   DF   12C7 13174         PUSHAL  REC_DSCR                ; Convert record to uppercase.
              00000310'EF   DF   12CD 13175         PUSHAL  REC_DSCR                ;
       00000000'GF     02   FB   12D3 13176         CALLS   #2,G^STR$UPCASE         ;
                                 12DA 13177
                                 12DA 13178 ;
                                 12DA 13179 ; Parse the record.
                                 12DA 13180 ;
                                 12DA 13181
000004CF'EF   000002EE'EF   B0   12DA 13182         MOVW    INRAB+RAB$W_RSZ,TPARSE_BLK+TPA$L_STRINGCNT
000004D3'EF   000002F4'EF   D0   12E5 13183         MOVL    INRAB+RAB$L_RBF,TPARSE_BLK+TPA$L_STRINGPTR
              00000000'EF   DF   12F0 13184         PUSHAL  CMD_KEY
              00000000'EF   DF   12F6 13185         PUSHAL  CMD_STATE
              000004C7'EF   DF   12FC 13186         PUSHAL  TPARSE_BLK
       00000000'GF     03   FB   1302 13187         CALLS   #3,G^LIB$TPARSE
                   A0 50    E8   1309 13188         BLBS    R0,10$                  ; Loop for entire file.
                                 130C 13189
                      06    E1   130C 13190 20$:    BBC     #MF$V_IN_OPEN,-         ; Branch if the file isn't open.
         2A 00000476'EF          130E 13191                 MISC_FLAGS,25$          ;
                   52  50   D0   1314 13192         MOVL    R0,R2                   ; Save error status.
                                 1317 13193         $CLOSE  FAB = INFAB             ; Close the file.
                   3A 50    E9   1324 13194         BLBC    R0,99$                  ; Branch on error.
00000476'EF   00000040 8F   CA   1327 13195         BICL    #MF$M_IN_OPEN,MISC_FLAGS; Indicate file is closed.
                   50  52   D0   1332 13196         MOVL    R2,R0                   ; Restore previous status.
       00000000'8F     50   D1   1335 13197         CMPL    R0,#RMS$_EOF            ; Was error condition END-OF-FILE?
                   11  13        133C 13198         BEQL    30$                     ; Branch if so.
       00000000'8F     50   D1   133E 13199 25$:    CMPL    R0,#RMS$_DEV            ; Was error bad or inappropriate device?
                   1A  12        1345 13200         BNEQ    99$                     ; Branch if not.
                      00    E1   1347 13201         BBC     #MF$V_INIT_FLAG,-       ; Ok if called by INIT, assume SHC$INIT
         12 00000476'EF          1349 13202                 MISC_FLAGS,99$          ;  not defined.
                   50  01   D0   134F 13203 30$:    MOVL    #SS$_NORMAL,R0          ; Indicate success.
                      00    E0   1352 13204         BBS     #MF$V_INIT_FLAG,-       ; Branch if initialization pass.
         07 00000476'EF          1354 13205                 MISC_FLAGS,99$          ;
       00002348'EF     00   FB   135A 13206         CALLS   #0,REFRESH              ; Update the screen.
                           04    1361 13207 99$:    RET
                                 1362 13208
                                 1362 13209
                                 1362 13210         .END    START
```

```
$$$AST                    = 00000000
$$$CNT                    = 00000003
$$$CURSIZ                 = 000001C4
$$$FLG                    = FFFFFFFF
$$$KEY                    = 0000004B
$$$KFG                    = FFFFFFFF
$$$MOD                    = 00000000
$$$NEWSIZ                 = 000001D0
$$$TMP                    = 000001E8 R     13
$$.TAB                    = 000002CC R     03
$$.TABEND                 = 00000310 R     03
$$.TMP                    = 00000000
$$.TMP1                   = 00000001
$$.TMP2                   = 000000CF
$$.TMPX                   = 00000004 R     04
$$.TMPX1                  = 0000000C
$$KEYTAB                  = 00000000 R     12
$$T1                      = 00000001
ACKR_SQ_ADRS                00000 2CA R    09
ACK_LM_ADRS                 00000329 R     09
ADD                         00000048 R     11
ADD_ALL_FIELDS              00000BCA RG    10
ADD_CIR                     00000260 R     11
ADD_CIR_QUAL                00000271 R     11
ADD_CIR_TYPE                0000028E R     11
ADD_CIR_TYPE_LOOP           0000029C R     11
ADD_CIR_TYPE_STRING         000002A8 R     11
ADD_CLUSTER                 000000BA R     11
ADD_CLUSTER_ALL             000000CB R     11
ADD_CON                     000002D3 R     11
ADD_CON_QUAL                000002E4 R     11
ADD_CON_TYPE                00000301 R     11
ADD_CON_TYPE_LOOP           0000030F R     11
ADD_CON_TYPE_STRING         0000031B R     11
ADD_COUNTERS                000000F6 R     11
ADD_COUNTERS_QUAL           00000107 R     11
ADD_CREDITS                 00000114 R     11
ADD_CREDITS_QUAL            00000125 R     11
ADD_ERRORS                  00000132 R     11
ADD_ERRORS_QUAL             00000143 R     11
ADD_FIELD                   00000A15 RG    10
ADD_ID                      00000CD5 RG    10
ADD_ID_LOOP                 000001B2 R     11
ADD_LOCAL_PORTS             00000150 R     11
ADD_LOCAL_PORTS_QUAL        00000161 R     11
ADD_MEMBERS                 000000D8 R     11
ADD_MEMBERS_QUAL            000000E9 R     11
ADD_NEXT_CLASS              0000009E R     11
ADD_NODE                    00000D3F RG    10
ADD_NODE_LOOP               00000246 R     11
ADD_ONE_SPACE               00002443 R     0F
ADD_SPACES                  00002449 R     0F
ADD_SYS                     0000016E R     11
ADD_SYS_ID                  000001A4 R     11
ADD_SYS_ID_STRING           000001BE R     11
ADD_SYS_NODE                00000238 R     11
ADD_SYS_NODE_STRING         00000252 R     11

ADD_SYS_QUAL                0000017F R     11
ADD_SYS_TYPE                000001CC R     11
ADD_SYS_TYPE_LOOP           000001DA R     11
ADD_SYS_TYPE_STRING         000001E6 R     11
ADD_TO_LIST                 00000DA0 R     10
ADRS_TABLE                  00002264 R     03
ASCIC                     = 00000004
ASCIC_DISPLAY               000026AF R     0F
ASCII                     = 00000001
ASCII_DISPLAY               000025C9 R     0F
ASC_TIM_DSCR                00001F8E R     03
AST_DEL                     00001A1E RG    0F
A_TIME                      000021B4 R     03
BDT_WAITS_ADRS              00000859 R     09
BEGIN_QUAL                  00000067 R     02
BITMAP_ADRS                 00002268 R     03
BLANK_LINES                 0000043C R     03
BLKS_REQ_ADRS               000007B5 R     09
BLKS_SENT_ADRS              00000793 R     09
BOTTOM_T                  = 00000076
BOT_CMD_LINE                00000471 R     03
BRDCST                      0000045B R     0F
BUFADR                      00000000 R     03
BUFFER_SIZE                 00002210 R     03
BUFF_DESCRIPTORS_ADRS       0000003B R     0E
BUFF_MSG                    000001DF R     02
BUFSIZE                     00000004 R     03
BUF_COUNT                   000003E4 R     03
BUF_QUO                     000004B5 R     03
BUILD_DECCRT_BOX            00001F9D R     0F
BUILD_VT52_BOX              00001F44 R     0F
CABLE_STATUS_ADRS           00000585 R     09
CB_STAT_STRG                0000028C R     02
CB_STAT_STRNG_LEN         = 00000007
CDT$C_LISTEN              = 00000001
CDT$C_OPEN                = 00000002
CDT$L_BYTMAPD             = 00000094
CDT$L_BYTMAPD_DSCR          000007FE R     09
CDT$L_BYTREQD             = 00000090
CDT$L_BYTREQD_DSCR          000007DE R     09
CDT$L_BYTSENT             = 00000088
CDT$L_BYTSENT_DSCR          000007BE R     09
CDT$L_CDTLST              = 0000006C
CDT$L_DGDISCARD           = 00000078
CDT$L_DGDISCARD_DSCR        00000715 R     09
CDT$L_DGRCVD              = 00000074
CDT$L_DGRCVD_DSCR           000006F4 R     09
CDT$L_DGSENT              = 00000070
CDT$L_DGSENT_DSCR           000006D3 R     09
CDT$L_LCONID              = 00000018
CDT$L_LCONID_DSCR           0000066D R     09
CDT$L_MSGINPUT            = 00000000
CDT$L_MSGRCVD             = 00000080
CDT$L_MSGRCVD_DSCR          00000759 R     09
CDT$L_MSGSENT             = 0000007C
CDT$L_MSGSENT_DSCR          00000737 R     09
CDT$L_RCONID              = 00000014
```

```
CDT$L_RCONID_DSCR                  0000068F R     09
CDT$L_REQDATS                    = 0000008C
CDT$L_REQDATS_DSCR                 0000079D R     09
CDT$L_SCSMSG                     = 0000002C
CDT$L_SNDDATS                    = 00000084
CDT$L_SNDDATS_DSCR                 0000077B R     09
CDT$W_BLKSTATE                   = 0000002A
CDT$W_BLKSTATE_DSCR                000006B1 R     09
CDT$W_INITLREC                   = 00000048
CDT$W_INITLREC_DSCR                000008E2 R     09
CDT$W_MINREC                     = 00000044
CDT$W_MINREC_DSCR                  00000903 R     09
CDT$W_MINSEND                    = 0000004A
CDT$W_MINSEND_DSCR                 00000880 R     09
CDT$W_PENDREC                    = 00000046
CDT$W_PENDREC_DSCR                 000008C1 R     09
CDT$W_QBDT_CNT                   = 0000009A
CDT$W_QBDT_CNT_DSCR                00000841 R     09
CDT$W_QCR_CNT                    = 00000098
CDT$W_QCR_CNT_DSCR                 00000820 R     09
CDT$W_REC                        = 00000042
CDT$W_REC_DSCR                     000008A1 R     09
CDT$W_SEND                       = 00000040
CDT$W_SEND_DSCR                    00000863 R     09
CDT$W_STATE                      = 00000028
CDT$W_STATE_DSCR                   0000064B R     09
CDT_BLKSTATE_STRNG                 000003C7 R     02
CDT_BUF_PNTR                       00000023 R     10
CDT_COUNT                          000003EB R     03
CDT_DSCR                           0000003F R     10
CDT_FIRST_LOOP                     00002FE5 R     0F
CDT_IN                           = 00000005
CDT_LOOP                           00002FAE R     0F
CDT_PFX                            0000049B R     02
CDT_SNAP                           00002F5A R     0F
CDT_SNAP_DONE                      00003009 R     0F
CDT_SNAP_END                       00003018 R     0F
CDT_STATE_STRNG                    0000035A R     02
CHAR_BUF_LEN                     = 0000000C
CHECK_ASCII                        000026E6 R     0F
CHECK_DIF                          00002D3D R     0F
CHECK_ID                           000014F8 R     0F
CHECK_MORE                         000018D4 R     0F
CHECK_NUM_CDT                      0000147A R     0F
CHECK_NUM_PB                       0000144C R     0F
CHECK_POS                          000014A8 R     0F
CHECK_QUE                          000032F4 R     0F
CHECK_VALID                        000026CF R     0F
CIBD$L_LINK                      = 0000000C
CIBDT$L_FREEBD                   = FFFFFFF4
CIR_STATE_ADRS                     00000541 R     09
CL$M_BRDCST                      = 00000001
CL$M_CMD                         = 00000002
CL$M_ERROR                       = 00000004
CL$M_SCROLL                      = 00000008
CL$V_BRDCST                      = 00000000
CL$V_CMD                         = 00000001
CL$V_ERROR                       = 00000002
CL$V_SCROLL                      = 00000003
CLASS                            = 00000100
CLASS_BOX_BUF                      0000076B R     03
CLASS_BOX_DSCR                     00000763 R     03
CLASS_BOX_LEN                    = 00000200
CLASS_BUF                          0000159B R     03
CLASS_BUF_LEN                    = 00000400
CLASS_DSCR                         00001593 R     03
CLASS_NAMES                        0000023B R     02
CLASS_SELECT                       00000001 R     10
CLEAR_SCREEN                       000023D0 R     0F
CLI$DCL_PARSE                      ********   X   0F
CLI$GET_VALUE                      ********   X   0F
CLI$PRESENT                        ********   X   0F
CLS$M_750_SYS                    = 00010000
CLS$M_780_SYS                    = 00020000
CLS$M_785_SYS                    = 00040000
CLS$M_ALL_CIR                    = 00000608
CLS$M_ALL_CON                    = 00001810
CLS$M_ALL_SYS                    = 00370002
CLS$M_ANY_CLUS                   = 0000007F
CLS$M_ANY_SCS                    = 0000007E
CLS$M_CIR                        = 00000008
CLS$M_CLUB                       = 00000001
CLS$M_CNT                        = 00000020
CLS$M_CON                        = 00000010
CLS$M_CRD                        = 00000040
CLS$M_CSB                        = 00000004
CLS$M_ERR                        = 00000100
CLS$M_HSC50_SYS                  = 00100000
CLS$M_HSC60_SYS                  = 00200000
CLS$M_LPRT                       = 00000080
CLS$M_NOOPEN_CIR                 = 00000400
CLS$M_NOOPEN_CON                 = 00001000
CLS$M_OPEN_CIR                   = 00000200
CLS$M_OPEN_CON                   = 00000800
CLS$M_SYS                        = 00000002
CLS$V_750_SYS                    = 00000010
CLS$V_780_SYS                    = 00000011
CLS$V_785_SYS                    = 00000012
CLS$V_790_SYS                    = 00000013
CLS$V_CIR                        = 00000003
CLS$V_CLUB                       = 00000000
CLS$V_CNT                        = 00000005
CLS$V_CON                        = 00000004
CLS$V_CRD                        = 00000006
CLS$V_CSB                        = 00000002
CLS$V_ERR                        = 00000008
CLS$V_HSC50_SYS                  = 00000014
CLS$V_HSC60_SYS                  = 00000015
CLS$V_LPRT                       = 00000007
CLS$V_NOOPEN_CIR                 = 0000000A
CLS$V_NOOPEN_CON                 = 0000000C
CLS$V_OPEN_CIR                   = 00000009
CLS$V_OPEN_CON                   = 0000000B
CLS$V_SYS                        = 00000001
```

```
CLSTR_H_SCROLL                   00000464 R     03     COUNT                          = 00000019
CLSTR_SCROLL_FLAGS               0000047C R     03     COUNTER_OWNER_ADRS               00000135 R     0E
CLU$GC_CLUB                      ********    X  0F     CROSS                          = 0000006E
CLUB$C_LENGTH                  = 000001A8              CR_WAITS_ADRS                    00000838 R     09
CLUB$L_FLAGS                   = 0000001C              CSB$B_ECOLVL                   = 00000040
CLUB$L_FLAGS_DSCR                00000063 R     09     CSB$B_ECOLVL_DSCR                00000330 R     09
CLUB$Q_FTIME                   = 0000002C              CSB$B_REMACKLIM                = 00000033
CLUB$Q_FTIME_DSCR                000000C5 R     09     CSB$B_REMACKLIM_DSCR             00000311 R     09
CLUB$Q_LST_TIME                = 0000003C              CSB$B_STATE                    = 00000043
CLUB$Q_LST_TIME_DSCR             000000E4 R     09     CSB$B_STATE_DSCR                 00000373 R     09
CLUB$T_QDNAME                  = 000000B8              CSB$B_UNACKEDMSGS              = 00000032
CLUB$T_QDNAME_DSCR               00000043 R     09     CSB$B_UNACKEDMSGS_DSCR           000002F1 R     09
CLUB$V_QF_VOTE                 = 00000019              CSB$B_VERNUM                   = 00000041
CLUB$W_MEMSEQ                  = 000000AC              CSB$B_WARMCDRPS                = 00000042
CLUB$W_MEMSEQ_DSCR               000000A6 R     09     CSB$B_WARMCDRPS_DSCR             00000351 R     09
CLUB$W_QDVOTES                 = 000000AE              CSB$C_LENGTH                   = 000000AC
CLUB$W_QDVOTES_DSCR              00000083 R     09     CSB$L_CSID                     = 0000004C
CLUB$W_QUORUM                  = 00000020              CSB$L_CSID_DSCR                  00000255 R     09
CLUB$W_QUORUM_DSCR               00000000 R     09     CSB$L_STATUS                   = 00000060
CLUB$W_VOTES                   = 00000022              CSB$L_STATUS1                  = 00000060
CLUB$W_VOTES_DSCR                00000022 R     09     CSB$L_STATUS1_DSCR               000003F2 R     09
CLUB_BUF                         0000004B R     10     CSB$L_STATUS2                  = 00000060
CLUB_HEADER_SIZE                 000021C8 R     03     CSB$L_STATUS2_DSCR               00000412 R     09
CLUB_IN                        = 00000001              CSB$L_STATUS_DSCR                00000434 R     09
CLUB_NUM_FIELDS                  00000427 R     03     CSB$M_LONG_BREAK               = 00000001
CLUSTER_BITMAP                   00000000 R     06     CSB$M_MEMBER                   = 00000002
CLUSTER_KEYTAB                   00000000 R     08     CSB$M_REMOVED                  = 00000004
CLUSTER_MAP_LEN                = 00000002              CSB$Q_REFTIME                  = 00000074
CLUSTER_MSG                      0000018A R     02     CSB$Q_REFTIME_DSCR               00000453 R     09
CLUSTER_SIZE_ARRAY               00000000 R     05     CSB$V_LONG_BREAK               = 00000000
CLUSTER_SNAP                     00002E2E R     0F     CSB$V_MEMBER                   = 00000001
CLUSTER_TABLE                    00000000 R     07     CSB$V_QF_ACTIVE                = 00000009
CLUSTER_TAB_LEN                = 00000118              CSB$V_QF_SAME                  = 00000003
CLUST_SAVE                       00002204 R     03     CSB$V_REMOVED                  = 00000002
CL_QDVOTES_ADRS                  0000009B R     09     CSB$W_ACKRSEQNM                = 00000030
CL_QUORUM_ADRS                   00000018 R     09     CSB$W_ACKRSEQNM_DSCR             000002B2 R     09
CL_SAVE_BITMAP                   0000000C R     06     CSB$W_LCKDIRWT                 = 00000054
CL_VOTES_ADRS                    0000003A R     09     CSB$W_LCKDIRWT_DSCR              000002D2 R     09
CMDS_QUEUED_ADRS                 00000231 R     0E     CSB$W_QDVOTES                  = 00000056
CMD_BUF                          000020DB R     03     CSB$W_QDVOTES_DSCR               000003D2 R     09
CMD_BUF_LEN                    = 00000078              CSB$W_QUORUM                   = 00000052
CMD_KEY                          00000000 RG    12     CSB$W_QUORUM_DSCR                000003B3 R     09
CMD_LINE                         0000225C R     03     CSB$W_RCVDSEQNM                = 0000002E
CMD_LINE_DSCR                    000020D3 R     03     CSB$W_RCVDSEQNM_DSCR             00000292 R     09
CMD_START                        00000000 R     11     CSB$W_SENDSEQNM                = 0000002C
CMD_STATE                        00000000 RG    11     CSB$W_SENDSEQNM_DSCR             00000272 R     09
CMD_STATUS                       000004B9 R     03     CSB$W_VOTES                    = 00000050
CNTRLC_AST_DEL                   00001A2A RG    0F     CSB$W_VOTES_DSCR                 00000395 R     09
CNVRT_ASCII                      000032D8 R     0F     CSB_BUF_PNTR                     0000001B R     10
CNX_STATE_ADRS                   0000038B R     09     CSB_COUNT                        000003E7 R     03
COL_NUM                          0000224C R     03     CSB_DSCR                         0000002F R     10
COMMON_SNAP                      000030D4 R     0F     CSB_IN                         = 00000003
CONTINUOUS_QUAL                  00000029 R     02     CSB_SNAP                         00002EB6 R     0F
CONVERT_DEC                      00002714 R     0F     CSB_SNAP_END                     00002EE0 R     0F
CONVERT_HEX                      000026FC R     0F     CSB_STATE_STRNG                  0000040B R     02
CONVERT_TO_HEX                   00000E5C R     10     CSB_STATE_STRNG_LEN            = 0000000A
CON_STATE_ADRS                   00000663 R     09     CSB_STATUS_STRNG                 00000452 R     02
```

```
CSB_STATUS_STRNG_LEN         = 00000007              DG_SIZE_ADRS                   000001E6 R     09
CSI                          = 0000009B              DIR_WT_ADRS                    000002EA R     09
CSID_ADRS                      0000026D R     09     DISPLAY                        00000FF9 R     0F
CTRLO                        = 00000017              DISPLAY_CABLE_STATUS           00002BB2 R     0F
CTRLZ                        = 0000001A              DISPLAY_CIR_STATE              00002B68 R     0F
CURRENT_CLASS                  000021F4 R     03     DISPLAY_CLUB                   000010A6 R     0F
CURRENT_REPORT                 00000000 R     10     DISPLAY_COMMON                 0000125A R     0F
CUR_PRIV                       00002230 R     03     DISPLAY_CON_STATE              00002C21 R     0F
CVT_TABLE                      000004A9 R     02     DISPLAY_CON_STATE_LEN        = 00000009
DATA_BUF                       000019A3 R     03     DISPLAY_CSB_STATE              00002AD0 R     0F
DATA_BUF_LEN                 = 00000400              DISPLAY_CSB_STATUS             00002AF2 R     0F
DATA_CNTRL                     000021F8 R     03     DISPLAY_END                    00001245 R     0F
DATA_DSCR                      0000199B R     03     DISPLAY_ERROR                  00001255 R     0F
DCS_TERM                     = 00000042              DISPLAY_FIELD                  00002989 R     0F
DCRT_DWN_ARROW               = 0000425B              DISPLAY_KBYTES                 00002C6F R     0F
DCRT_LFT_ARROW               = 0000435B              DISPLAY_LOOPBACK_STATUS        00002CE9 R     0F
DCRT_NARROW_DSCR               0000014E R     02     DISPLAY_PORT_STATUS            00002C91 R     0F
DCRT_NOATR_DSCR                00000143 R     02     DISPLAY_PORT_TYPE              00002CB8 R     0F
DCRT_RT_ARROW                = 0000445B              DISPLAY_PROC_NAME              00002BF4 R     0F
DCRT_UP_ARROW                = 0000415B              DISPLAY_PROTOCOL               00002A96 R     0F
DCRT_WIDE_DSCR                 0000015B R     02     DISPLAY_PTP                    00001170 R     0F
DDB$L_UCB                    = 00000004              DISPLAY_QF                     00002A44 R     0F
DDB$T_NAME                   = 00000014              DISPLAY_QF_ACTIVE              00002A2E R     0F
DDB_LOOP                       000030F4 R     0F     DISPLAY_QF_SAME                00002A18 R     0F
DDM_SIZE                       000020D1 R     03     DISPLAY_QF_VOTE                00002A02 R     0F
DEC                          = 00000003              DISPLAY_RPORT_TYP              00002B19 R     0F
DECIMAL_ID                     00000E93 R     10     DISPLAY_RSTATE                 00002B92 R     0F
DEC_DISPLAY                    00002653 R     0F     DISPLAY_SCS                    00001109 R     0F
DEC_H_SCROLL                   0000188A R     0F     DISPLAY_SCS_STATE              00002C48 R     0F
DEC_SIZE                     = 00000003              DISPLAY_SCS_STATE_LEN        = 00000009
DEC_TAB                        000004CA R     02     DISPLAY_SOFTWARE               00002A5D R     0F
DEC_TAB_END                    0000056A R     02     DISPLAY_STRING                 000029E6 R     0F
DEC_TAB_LEN                  = 00000014              DO_SNAPSHOT                    0000052E R     0F
DEC_V_SCROLL                   00001748 R     0F     DRAW_LAST_LINE                 000020F1 R     0F
DEF_CONNCNT                  = 00000012              DRAW_LINE                      00002466 R     0F
DEF_EXT                        00000001 R     02     DT$_CI750                    = 00000002
DEF_EXT_LEN                  = 00000004              DT$_CI780                    = 00000001
DEF_NAME2                      00000005 R     02     DT$_VT5X                     = 00000040
DEF_NAME2_LEN                = 00000016              DUMMY_PB                       00000253 R     10
DELAY                          000021AC R     03     DUMMY_SB                       000001F3 R     10
DELM                           0000241F R     0F     DVI$_DEVBUFSIZ               = 00000008
DELM_DSCR                      00000173 R     02     DVI$_DEVCLASS                = 00000004
DELM_DSCR_LEN                = 00000003              DVI$_DEVDEPEND               = 0000000A
DEV$V_NET                      ********   X   0F     DVI$_DEVDEPEND2              = 0000001C
DEV$V_TRM                      ********   X   0F     DVI$_DEVTYPE                 = 00000006
DEVDEPEND                      00002218 R     03     DVT$M_AVO                    = 00000010
DEVDEPEND2                     0000221C R     03     DVT$M_DCRT                   = 00000002
DEVICE_CLASS                   00002208 R     03     DVT$M_DISK                   = 00000020
DEVICE_TYPE                    0000220C R     03     DVT$M_HARD                   = 00000004
DEV_TYP_STRNG                  000002F2 R     02     DVT$M_UNKNOWN                = 00000008
DEV_TYP_STRNG_LEN            = 00000005              DVT$M_VT52                   = 00000001
DGI_MAP_ADRS                   000002BA R     0E     DVT$V_AVO                    = 00000004
DGS_DSCRD_ADRS                 0000072D R     09     DVT$V_DCRT                   = 00000001
DGS_FREE_ADRS                  000000F2 R     0E     DVT$V_DISK                   = 00000005
DGS_RCVD_ADRS                  0000070C R     09     DVT$V_HARD                   = 00000002
DGS_SENT_ADRS                  000006EB R     09     DVT$V_UNKNOWN                = 00000003
DG_OVRHD_SIZE_ADRS             000001A5 R     0E     DVT$V_VT52                   = 00000000
```

```
ENDING_QUAL                      00000059 R     02
ERASE_LINE                       00002402 R     0F
ERASE_PAGE                       00001A99 R     0F
ERASE_TWO_LINES                  000023FD R     0F
ERR_MSGVEC                       0000049D R     03
ERR_MSG_DSCR                     00001E7E R     03
ERT_COUNT_ADRS                   0000031C R     0E
ERT_MAX_ADRS                     0000033E R     0E
ESC                            = 0000001B
EVF$M_AST_FLAG                 = 00000004
EVF$M_TIM_FLAG                 = 00000002
EVF$V_AST_FLAG                 = 00000002
EVF$V_TIM_FLAG                 = 00000001
EXCLUDE_CLASS                    00000C81 RG    10
EXE$GL_LOCKRTRY                  ********  X    0F
EXECUTE                          00001215 RG    10
EXIT_BLK                         00000481 R     03
EXIT_RTN                         00000F39 RG    0F
FAB$B_DNS                      = 00000035
FAB$C_BID                      = 00000003
FAB$C_BLN                      = 00000050
FAB$C_SEQ                      = 00000000
FAB$C_VAR                      = 00000002
FAB$L_ALQ                      = 00000010
FAB$L_DEV                      = 00000040
FAB$L_DNA                      = 00000030
FAB$L_FOP                      = 00000004
FAB$V_CHAN_MODE                = 00000002
FAB$V_FILE_MODE                = 00000004
FAB$V_LNM_MODE                 = 00000000
FAB$V_NAM                      = 00000018
FAB$V_PUT                      = 00000000
FAB$W_GBC                      = 00000048
FD$B_FIELD_SIZE                  00000000
FD$B_FORMAT                      00000005
FD$B_LENGTH                      00000004
FD$B_MAX_DATA_SIZE               00000002
FD$B_MAX_FLD_SIZE                00000003
FD$B_MIN_FLD_SIZE                00000001
FD$B_TITLE_SIZE                  00000018
FD$L_CLASS                       0000000C
FD$L_DISPLAY_ROUTINE             00000008
FD$L_OFFSET                      00000010
FD$L_OFFSET_PRFX                 00000014
FD$T_TITLE                       00000019
FDONE                            000026C7 R     0F
FERR                             000026CA R     0F
FIELD_BOT_BOX_BUF                00000B7B R     03
FIELD_BOT_BOX_DSCR               00000B73 R     03
FIELD_BOT_BOX_LEN              = 00000200
FIELD_TOP_BOX_BUF                00000973 R     03
FIELD_TOP_BOX_DSCR               0000096B R     03
FIELD_TOP_BOX_LEN              = 00000200
FIND_ENTRY                       00002363 R     0F
FIND_NODE                        00000D5B R     10
FIND_SYS_ID                      00000E16 R     10
FIRST_DATA_LINE                  00002254 R     03
FIRST_LINE                       00002250 R     03
FIX_BUF                          0000004B R     10
FIX_BUF_LEN                    = 00000458
FLAG_STATE                       00000472 R     03
FORMAT_BUF                       0000215B R     03
FORMAT_BUF_DSCR                  00002153 R     03
FORMAT_BUF_LEN                   00002153 R     03
FORMAT_DATA                      0000257F R     0F
FORMED_ADRS                      000000DD R     09
FORM_CIRCS_ADRS                  0000020E R     0E
FORM_PB_DONE                     000031D2 R     0F
FORM_PB_LOOP                     00003140 R     0F
FORM_PB_SNAP                     0000311E R     0F
FREE_BUFF_DESCRIPTORS_ADRS       00000064 R     0E
FREE_BUFF_MSG                    00000205 R     02
FREE_BUF_END                     000032B8 R     0F
FREE_BUF_LOOP                    000032AE R     0F
FREE_BUF_SNAP                    0000329E R     0F
GET_BLANK                        00002520 R     0F
GET_CLASSES                      000022A0 R     0F
GET_DATA                         00002729 RG    0F
GET_HEADER                       00001C46 R     0F
GET_HELP                         00001077 RG    10
GET_LIST                         0000057E R     02
GET_MSG_BUF                      00001E8E R     03
GET_MSG_BUF_LEN                = 00000100
GET_MSG_DSCR                     00001E86 R     03
GET_NODE_NAME                    00000F1B RG    0F
GET_NUM_PB                       000008FB R     10
GET_TITLES                       00002144 R     0F
GET_VAL_DSCR                     00001DA3 R     03
GRAPHICS_DSGNTR                  00000168 R     02
GRAPHICS_OFF                   = 0000000F
GRAPHICS_ON                    = 0000000E
HEADER_END                     = 00000007
HEADER_SIZE                      000021C4 R     03
HEADER_START                   = 00000002
HELP                             0000064A R     11
HELP_DSCR                        00000553 R     03
HELP_FLAGS                       0000047D R     03
HEX                            = 00000002
HEX_DISPLAY                      000025F9 R     0F
HIGH_NYBLE                       0000262C R     0F
HLP$M_HELP                     = 00000020
HLP$M_PROMPT                   = 00000001
HW_TYPE_ADRS                     00000180 R     09
HW_TYPE_TABLE                    000004B2 R     02
HW_VERS_ADRS                     00000160 R     09
H_SCROLL_LINE                    0000046C R     03
ID_COUNT                         000022D8 R     03
ID_LIST_ENTRYS                 = 00000010
ID_LIST_LEN                    = 00000060
ID_RESULT                        000022DC R     03
INAM_BLOCK                       0000006C R     03
INCARNATION_ADRS                 00000227 R     09
INCLUDE_CLASS                    00000C2D RG    10
INCN_TIME_ADRS                   0000024B R     09
```

INC_H_SCROLL		0000185D	R	0F	LB_STAT_STRG_LEN	=	00000005		
INC_V_SCROLL		000017C5	R	0F	LEFT_T	=	00000074		
INFAB		0000001C	R	03	LEN	=	00000009		
INIT		00000862	R	0F	LIB$ASN_WTH_MBX		********	X	0F
INIT_NAME		0000012A	R	02	LIB$CVT_DTB		********	X	0F
INIT_REC_ADRS		000008FA	R	09	LIB$CVT_TIME		********	X	0F
INRAB		000002CC	R	03	LIB$DATE_TIME		********	X	0F
INSTRNG_LEN		000004C5	R	03	LIB$DOWN_SCROLL		********	X	0F
INTERLOCK_SNAP		00003249	R	0F	LIB$ERASE_LINE		********	X	0F
INTERVAL_QUAL		0000003B	R	02	LIB$ERASE_PAGE		********	X	0F
INVRS_DSCR		0000017E	R	02	LIB$FREE_VM		********	X	0F
IN_CHAN		00002234	R	03	LIB$GET_INPUT		********	X	0F
IN_DEV		00002246	R	03	LIB$GET_VM		********	X	0F
IN_DEV_NAME		000000DE	R	02	LIB$LOOKUP_KEY		********	X	10
IN_EXP_DSCR		0000011C	R	03	LIB$PUT_BUFFER		********	X	0F
IN_EXP_NAME		00000124	R	03	LIB$PUT_OUTPUT		********	X	10
IN_EXP_NAME_LEN	=	00000050			LIB$PUT_SCREEN		********	X	0F
IN_FILE_SPEC		000000CC	R	03	LIB$SET_BUFFER		********	X	0F
IN_FILE_SPEC_LEN	=	00000050			LIB$SET_CURSOR		********	X	0F
IO$M_CTRLCAST		********	X	0F	LIB$SET_OUTPUT		********	X	0F
IO$M_DSABLMBX		********	X	0F	LIB$SET_SCROLL		********	X	0F
IO$M_ENABLMBX		********	X	0F	LIB$SIGNAL		********	X	0F
IO$M_EXTEND		********	X	0F	LIB$TPARSE		********	X	0F
IO$M_TYPEAHDCNT		********	X	0F	LIB$UP_SCROLL		********	X	0F
IO$_READPROMPT		********	X	0F	LIB$_AMBKEY		********	X	0F
IO$_READVBLK		********	X	0F	LIB$_SYNTAXERR		********	X	0F
IO$_SENSEMODE		********	X	0F	LIB$_UNRKEY		********	X	0F
IO$_SETMODE		********	X	0F	LIBRARY_NAME		0000010C	R	02
IO$_WRITEVBLK		********	X	0F	LINE_DIF		00000434	R	03
IOC$GL_DEVLIST		********	X	0F	LINE_NUM		00002248	R	03
IOSB		0000223E	R	03	LISTENER_SNAP		00003030	R	0F
IPL$_SCS	=	00000008			LISTENER_SNAP_END		000030CD	R	0F
JPI$_CURPRIV		********	X	02	LL_CORNER	=	0000006D		
JPI_LIST		000005BE	R	02	LOCAL_BITMAP		00000000	R	0B
KB_MAPPED_ADRS		00000816	R	09	LOCAL_H_SCROLL		00000468	R	03
KB_RCVD_ADRS		000007F6	R	09	LOCAL_KEYTAB		00000000	R	0D
KB_SENT_ADRS		000007D6	R	09	LOCAL_MAP_LEN	=	00000000		
KEYTAB_ADRS		00002274	R	03	LOCAL_MSG		000001A6	R	02
KEYWRDVAL	=	00000019			LOCAL_REPORT		00001513	R	0F
KEY_DSCR		0000054B	R	03	LOCAL_SAVE		00002200	R	03
KEY_MSK		00000547	R	03	LOCAL_SCROLL_FLAGS		0000047B	R	03
KEY_OFF		00000543	R	03	LOCAL_SIZE_ARRAY		00000000	R	0A
KEY_VAL		0000053F	R	03	LOCAL_SNAP		000030DF	R	0F
LAST_BRDCST_BUF		00001DFA	R	03	LOCAL_TABLE		00000000	R	0C
LAST_BRDCST_DSCR		00001DF2	R	03	LOCAL_TAB_LEN	=	00000064		
LAST_CLUB_LINE_BUF		00000F8B	R	03	LOCK_FAIL		0000332F	R	0F
LAST_CLUB_LINE_DSCR		00000F83	R	03	LOCSYSID		00000005	R	10
LAST_CLUB_LINE_LEN	=	00000200			LOCSYSID_DSCR		00000000	R	0E
LAST_DATA_LINE		00002258	R	03	LOC_CONID_ADRS		00000685	R	09
LAST_LINE		00002260	R	03	LOC_PROC_NAME_ADRS		00000617	R	09
LAST_LINE_BUF		00000D83	R	03	LOC_SAVE_BITMAP		00000004	R	0B
LAST_LINE_DSCR		00000D7B	R	03	LOC_SB_ADRS		00000013	R	10
LAST_LINE_LEN	=	00000200			LOG_MAP_ADRS		000002DA	R	0E
LAST_TRANSITION_ADRS		000000FC	R	09	LOOKUP_FIELD		00000B3A	RG	10
LBR$OUTPUT_HELP		********	X	10	LOOP		00003318	R	0F
LB_STATUS_ADRS		000002FA	R	0E	LOWERIPL		0000332C	R	0F
LB_STAT_STRG		000002AC	R	02	LOW_NYBLE		00002638	R	0F

```
LPORT_NAME_ADRS                 00000493 R    09     MF$M_NEW_CHAR                 = 00000080
LP_STATUS_ADRS                  000000AF R    0E     MF$M_RIP                      = 00000008
LP_TYPE_ADRS                    000001CB R    0E     MF$M_SCROLL                   = 00000100
LR_CORNER                     = 0000006A             MF$M_VALID_MSG                = 00000200
LSYSTEM_ID_ADRS                 00000018 R    0E     MF$V_CONT                     = 00000004
MAIN_CONT                       0000018D R    0F     MF$V_CR_FOUND                 = 0000000A
MAIN_ERR                        00000F42 R    0F     MF$V_CYCLE                    = 00000005
MAIN_EXIT                       00000FE8 R    0F     MF$V_EXIT_FLAG                = 00000001
MAIN_LOOP                       00000012 R    0F     MF$V_INIT_FLAG                = 00000000
MAP_LEN                         0000226C R    03     MF$V_INVERSE                  = 00000002
MAP_LENGTH                    = 00000000             MF$V_IN_OPEN                  = 00000006
MASK                          = 01C060F8             MF$V_NEW_CHAR                 = 00000007
MASK_BIT                      = 02000000             MF$V_RIP                      = 00000003
MAXPDT                        = 00000002             MF$V_SCROLL                   = 00000008
MAX_ID                        = 00000010             MF$V_VALID_MSG                = 00000009
MAX_MSG                         000004B1 R    03     MIN_REC_ADRS                    0000091B R    09
MAX_PORT_ADRS                   00000279 R    0E     MIN_SEND_ADRS                   00000898 R    09
MB$B_ERTCNT                     00000010             MISC_BUF                        00000447 R    10
MB$B_ERTCNT_DSCR                00000304 R    0E     MISC_BUF_LEN                  = 0000002E
MB$B_ERTMAX                     00000011             MISC_FLAGS                      00000476 R    03
MB$B_ERTMAX_DSCR                00000326 R    0E     MORE_DSCR                       00000078 R    02
MB$L_DEVTYPE                    00000008             MORE_DWN                        000000B2 R    02
MB$L_DEVTYPE_DSCR               000001B3 R    0E     MORE_LFT                        000000BD R    02
MB$L_FORM_PB                    00000014             MORE_RT                         000000C8 R    02
MB$L_FORM_PB_DSCR               000001F6 R    0E     MORE_TABS                       00000090 R    02
MB$L_PL_WAITERS                 00000018             MORE_UP                         000000A7 R    02
MB$L_PL_WAITERS_DSCR            00000143 R    0E     MSG$_TRMBRDCST                = 00000053
MB$L_PORT_NAME                  00000000             MSG$_TRMUNSOLIC               = 00000001
MB$L_PORT_NAME_DSCR             0000007A R    0E     MSG1_DSCR                       000001BC R    02
MB$L_PORT_NUM                   00000004             MSG2_DSCR                       000001CF R    02
MB$L_PORT_NUM_DSCR              000000B9 R    0E     MSGS_FREE_ADRS                  00000113 R    0E
MB$L_STS                        0000000C             MSGS_RCVD_ADRS                  00000771 R    09
MB$L_STS_DSCR                   00000097 R    0E     MSGS_SENT_ADRS                  0000074F R    09
MB$W_CMDS_QUED                  00000020             MSG_FRE_QUE                     00000008 R    03
MB$W_CMDS_QUED_DSCR             00000219 R    0E     MSG_HDR_SIZE_ADRS               00000180 R    0E
MB$W_DG_FREE                    0000001C             MSG_OUT_ADRS                    000004AD R    03
MB$W_DG_FREE_DSCR               000000DA R    0E     MSG_OUT_QUE                     00000010 R    03
MB$W_ERRCNT                     00000012             MSG_SIZE_ADRS                   00000206 R    09
MB$W_ERRCNT_DSCR                00000346 R    0E     NAM$B_ESS                     = 0000000A
MB$W_MSG_FREE                   0000001E             NAM$B_NOP                     = 00000008
MB$W_MSG_FREE_DSCR              000000FB R    0E     NAM$B_RSS                     = 00000002
MB$W_RSPS_QUED                  00000022             NAM$C_BID                     = 00000002
MB$W_RSPS_QUED_DSCR             0000023D R    0E     NAM$C_BLN                     = 00000060
MBXCHAN                         0000223C R    03     NAM$L_ESA                     = 0000000C
MBX_BUF                         00001FB5 R    03     NAM$L_RSA                     = 00000004
MBX_BUF_LEN                   = 00000100             NAM$T_DVI                     = 00000014
MBX_MSG_DSCR                    00001FAD R    03     NAME_ADRS                       00000092 R    0E
MB_IN                         = 00000007             NARROW                        = 00000050
MEMSEQ_ADRS                     000000BE R    09     NEW_CHAR_BUF                    00002220 R    03
MESSAGE                         00002FF8 R    0F     NEXT_ADD_CIR_QUAL               00000286 R    11
MF$M_CONT                     = 00000010             NEXT_ADD_CON_QUAL               000002F9 R    11
MF$M_CR_FOUND                 = 00000400             NEXT_ADD_SYS_QUAL               0000019C R    11
MF$M_CYCLE                    = 00000020             NEXT_REM_CIR_QUAL               00000509 R    11
MF$M_EXIT_FLAG                = 00000002             NEXT_REM_CON_QUAL               0000056F R    11
MF$M_INIT_FLAG                = 00000001             NEXT_REM_SYS_QUAL               0000042C R    11
MF$M_INVERSE                  = 00000004             NEXT_SET_FIELD_QUAL             000005D8 R    11
MF$M_IN_OPEN                  = 00000040             NODE_ADRS                       00000124 R    09
```

```
NODE_NAME                         00000450 R     03
NODE_NAME_DSCR                    00000448 R     03
NO_MESSAGE                        00003001 R     0F
NO_MORE                           000000D3 R     02
NUCL_DEV_NAME                     00000101 R     02
NUM_BUF_DSCR                      0000000B R     10
NUM_BUF_DSCR_DSCR                 00000023 R     0E
NUM_CIRCUITS_ADRS                 000001C1 R     09
NUM_CLASSES                     = 00000009
NUM_CLUSTER_DSCRS               = 00000046
NUM_CONNECTIONS_ADRS              00000519 R     09
NUM_ERRORS_ADRS                   0000035E R     0E
NUM_FIELDS                        00000425 R     03
NUM_FREE_BUF_DSCR                 0000000F R     10
NUM_FREE_BUF_DSCR_DSCR            0000004C R     0E
NUM_HW_TYPES                    = 00000006
NUM_LINES                         00000428 R     03
NUM_LOCAL_DSCRS                 = 00000019
OLD_CDT_DSCR                      00000411 R     03
OLD_CSB_DSCR                      00000401 R     03
OLD_FIX_BUF                       000004A3 R     10
OLD_PB_DSCR                       00000409 R     03
OLD_SB_DSCR                       000003F9 R     03
OLD_VAR_BUF                       00000419 R     03
OLD_VAR_BUF_END                   0000041D R     03
OLD_VAR_BUF_LEN                   00000421 R     03
OPEN_CIRCS_ADRS                   000001EB R     0E
OTS$CVT_TZ_L                      ********   X   10
OUTFAB                            00000174 R     03
OUTNAM_BLOCK                      000001C4 R     03
OUTPUT_DDM                        00001D97 R     0F
OUTPUT_QUAL                       0000004B R     02
OUTPUT_TIME                       00001E62 R     0F
OUTPUT_WIDTH                      0000222C R     03
OUT_CHAN                          00002236 R     03
OUT_DEV                           00002247 R     03
OUT_DEV_NAME                      000000EF R     02
OUT_EXP_DSCR                      00000274 R     03
OUT_EXP_NAME                      0000027C R     03
OUT_EXP_NAME_LEN                = 00000050
OUT_FILE_SPEC                     00000224 R     03
OUT_FILE_SPEC_LEN               = 00000050
OUT_MSG                           0000161A R     0F
PAAC                            = 41415003
PABC                            = 42415003
PARSE_INTERVAL                    00000624 R     11
PARSE_SCREEN                      0000062E R     11
PARSE_VALUE                       00000638 R     11
PB$B_CBL_STS                    = 00000028
PB$B_CBL_STS_DSCR                 0000056D R     09
PB$B_P0_STS                     = 00000029
PB$B_P1_STS                     = 0000002A
PB$B_RSTATE                     = 00000021
PB$B_RSTATE_DSCR                  0000054B R     09
PB$B_RSTATION                   = 0000000C
PB$B_RSTATION_DSCR                0000049E R     09
PB$B_RST_PORT                   = 00000020
PB$B_RST_PORT_DSCR                000004C0 R     09
PB$C_LENGTH                     = 00000054
PB$C_OPEN                       = 00000003
PB$C_VC_FAIL                    = 00008000
PB$L_BLINK                      = 00000004
PB$L_CDTLST                     = 00000034
PB$L_RPORT_FCN                  = 0000001C
PB$L_RPORT_FCN_DSCR               000005B6 R     09
PB$L_RPORT_REV                  = 00000018
PB$L_RPORT_REV_DSCR               00000592 R     09
PB$L_RPORT_TYP                  = 00000014
PB$L_RPORT_TYP_DSCR               000004E1 R     09
PB$L_SBLINK                     = 00000030
PB$L_WAITQFL                    = 00000038
PB$L_WAITQFL_DSCR                 000005DB R     09
PB$M_CUR_CBL                    = 00000001
PB$M_CUR_PS                     = 00000001
PB$SHC$L_NUM_CDT                = 00000004
PB$SHC$L_NUM_CDT_DSCR             00000501 R     09
PB$S_PORT_TYP                   = 0000001F
PB$T_LPORT_NAME                 = 00000024
PB$T_LPORT_NAME_DSCR              0000047B R     09
PB$V_PORT_TYP                   = 00000000
PB$W_STATE                      = 00000012
PB$W_STATE_DSCR                   00000529 R     09
PB_BUF_PNTR                       0000001F R     10
PB_COUNT                          000003E9 R     03
PB_DSCR                           00000037 R     10
PB_FIRST_LOOP                     00002F25 R     0F
PB_IN                           = 00000004
PB_LOOP                           00002F0B R     0F
PB_LOOP_END                       0000301B R     0F
PB_RSTATE_STRNG                   00000330 R     02
PB_RSTATE_STRNG_LEN             = 00000008
PB_SNAP                           00002EE4 R     0F
PB_STATE_STRNG                    0000030D R     02
PB_STATE_STRNG_LEN              = 00000007
PDT$B_DQIMAP                      00000154
PDT$B_DQIMAP_DSCR                 000002A2 R     0E
PDT$B_HSHUT_DG                    000001B0
PDT$B_MAX_PORT                    0000017C
PDT$B_MAX_PORT_DSCR               00000261 R     0E
PDT$B_NXT_PORT                    0000017E
PDT$B_P0_LBSTS                    00000180
PDT$B_P0_LBSTS_DSCR               000002E2 R     0E
PDT$B_P1_LBSTS                    00000181
PDT$B_PLOGMAP                     00000134
PDT$B_PLOGMAP_DSCR                000002C2 R     0E
PDT$B_PORTMAP                     00000114
PDT$B_PORTMAP_DSCR                00000282 R     0E
PDT$B_PORT_NUM                    0000017D
PDT$B_REQIDPS                     0000017F
PDT$C_LENGTH                    = 000000E4
PDT$C_PAREGBASE                   000000E4
PDT$C_PAREGEND                    00000110
PDT$C_PQB                       = 000001E0
PDT$L_CNF                         000000E4
```

```
PDT$L_CQ0                       000000F0
PDT$L_CQ1                       000000F4
PDT$L_DFQ                       000000FC
PDT$L_DFQHDR                    00000208
PDT$L_DGHDRSZ                   00000190
PDT$L_DGNETHD                   00000194
PDT$L_DGOVRHD                 = 000000B8
PDT$L_DGOVRHD_DSCR              0000018D R      0E
PDT$L_DQELOGOUT                 000002E0
PDT$L_GPTBASE                   0000022C
PDT$L_GPTLEN                    00000230
PDT$L_LBDG                      00000184
PDT$L_MFQ                       00000100
PDT$L_MFQHDR                    0000020C
PDT$L_MQELOGOUT                 00000320
PDT$L_MSGHDRSZ                = 000000B4
PDT$L_MSGHDRSZ_DSCR             00000168 R      0E
PDT$L_MTC                       00000104
PDT$L_PFAR                      00000108
PDT$L_PMC                       000000E8
PDT$L_POLLERDUE                 0000018C
PDT$L_POOLDUE                   00000188
PDT$L_PPR                       0000010C
PDT$L_PS                        000000EC
PDT$L_PSR                       000000F8
PDT$L_SPTBASE                   00000224
PDT$L_SPTLEN                    00000228
PDT$L_VBDT                      0000021C
PDT$L_VPQB                      00000218
PDT$L_WAITQFL                 = 000000AC
PDT$M_PRV_LBS                 = 00000002
PDT$Q_COMQ2                     000001F0
PDT$Q_COMQ3                     000001F8
PDT$Q_COMQBASE                  000001E0
PDT$Q_COMQH                     000001E8
PDT$Q_COMQL                     000001E0
PDT$Q_DFREEQ                    000001D0
PDT$Q_FORMPB                    00000174
PDT$Q_MFREEQ                    000001D8
PDT$Q_RSPQ                      00000200
PDT$Q_TEMP_RSPQ                 0000019C
PDT$T_CNTOWNER                = 000000C4
PDT$T_CNTOWNER_DSCR             0000011D R      0E
PDT$V_LBDG                    = 00000002
PDT$W_BDTLEN                    00000220
PDT$W_DQELEN                    00000210
PDT$W_LPORT_STS                 00000110
PDT$W_MQELEN                    00000214
PDT$W_PBCOUNT                   00000112
PDT$W_PBCOUNT_DSCR              000001D3 R      0E
PDT$W_SIZE                    = 00000008
PDT$W_STDGDYN                   00000198
PDT$W_STDGUSED                  0000019A
PDT_BUF                         000002A7 R      10
PDT_IN                        = 00000006
PDT_PFX                         000004A0 R      02
PEND_REC_ADRS                   000008D9 R      09
PFX_LIM                       = 00000007
POOL_WAITERS_ADRS               0000015B R      0E
POOL_WAITER_LOOP                0000323C R      0F
POOL_WAITER_LOOP_END            00003249 R      0F
PORT_FORM_PB_LOOP               0000320F R      0F
PORT_FORM_PB_LOOP_END           0000321C R      0F
PORT_NUM_ADRS                   000000D1 R      0E
PORT_SNAP                       000031D9 R      0F
PORT_SNAP_END                   000032D1 R      0F
PORT_STATUS_STRNG               000002FE R      02
PORT_STATUS_STRNG_LEN         = 00000007
PPD$B_DEF_ST                    0000001C
PPD$B_FLAGS                     0000000F
PPD$B_HWVERS                    00000034
PPD$B_LBDATA                    00000012
PPD$B_LCB_0                     00000012
PPD$B_LCB_LPORT                 00000010
PPD$B_LCB_NPORT                 0000000F
PPD$B_LCB_OPC                   00000011
PPD$B_LCB_PORT                  0000000E
PPD$B_OPC                       0000000E
PPD$B_PORT                      0000000C
PPD$B_PROTOCOL                  0000001A
PPD$B_RSTATE                    00000025
PPD$B_RST_PORT                  00000024
PPD$B_STATUS                    0000000D
PPD$B_SWFLAG                    0000000B
PPD$B_SYSTEMID                  00000014
PPD$B_TYPE                      0000000A
PPD$C_LB_LENGTH                 00000046
PPD$C_LCB_DATA                  00000013
PPD$C_LENGTH                    00000012
PPD$C_MIN_DGSIZ                 00000050
PPD$K_LB_LENGTH                 00000046
PPD$K_LENGTH                    00000012
PPD$L_BLINK                     00000004
PPD$L_DG_DISC                   00000028
PPD$L_FLINK                     00000000
PPD$L_IN_VCD                    00000018
PPD$L_LBCRC                     00000042
PPD$L_P0_ACK                    00000010
PPD$L_P0_NAK                    00000014
PPD$L_P0_NRSP                   00000018
PPD$L_P1_ACK                    0000001C
PPD$L_P1_NAK                    00000020
PPD$L_P1_NRSP                   00000024
PPD$L_REC_BOFF                  00000028
PPD$L_REC_NAME                  00000024
PPD$L_RPORT_FCN                 00000020
PPD$L_RPORT_REV                 0000001C
PPD$L_RPORT_TYP                 00000018
PPD$L_SND_BOFF                  00000020
PPD$L_SND_NAME                  0000001C
PPD$L_ST_ADDR                   00000018
PPD$L_XCT_LEN                   00000018
PPD$Q_CURTIME                   00000048
PPD$Q_NODENAME                  00000040
```

```
PPD$Q_SWINCARN                    00000028
PPD$Q_XCT_ID                      00000010
PPD$T_HWTYPE                      00000030
PPD$T_SWTYPE                      00000020
PPD$T_SWVERS                      00000024
PPD$W_LCB_LEN7                    0000000C
PPD$W_LENGTH                      00000010
PPD$W_MASK                        00000010
PPD$W_MAXDG                       0000001C
PPD$W_MAXMSG                      0000001E
PPD$W_MTYPE                       00000012
PPD$W_M_VAL                       00000014
PPD$W_SIZE                        00000008
PR$_IPL                           ********     X    0F
PRIOR_BLANK_LINES                 00000440 R        03
PRIOR_LINE_DIF                    00000438 R        03
PRIOR_NUM_LINES                   0000042C R        03
PROMPT                            00000229 R        02
PROMPT_LEN                      = 0000000A
PROTOCOL_ADRS                     00000348 R        09
PRT_MAP_ADRS                      0000029A R        0E
PRV$V_CMKRNL                    = 00000000
QDVOTES_ADRS                      000003EA R        09
QD_NAME_ADRS                      0000005B R        09
QF_ACTIVE_ADRS                    0000042A R        09
QF_SAME_ADRS                      0000040A R        09
QF_VOTE_ADRS                      0000007B R        09
QUE_LIMIT                       = 00000003
QUORUM_ADRS                       000003CB R        09
RAB$B_RAC                       = 0000001E
RAB$C_BID                       = 00000001
RAB$C_BLN                       = 00000044
RAB$C_SEQ                       = 00000000
RAB$L_CTX                       = 00000018
RAB$L_RBF                       = 00000028
RAB$L_ROP                       = 00000004
RAB$W_RSZ                       = 00000022
RCVD_SQ_ADRS                      000002AA R        09
READ_LOOP                         00000199 R        0F
READ_WO_PROMPT                    000001F7 R        0F
READ_W_PROMPT                     00000295 R        0F
RECEIVE_ADRS                      000008B9 R        09
REC_BUFF                          00000318 R        03
REC_DSCR                          00000310 R        03
REC_SIZE                        = 00000084
REFRESH                           00002348 RG       0F
REMOVE                            00000346 R        11
REMOVE_FIELD                      00000AC6 RG       10
REMOVE_FROM_LIST                  00000DFE R        10
REM_CIR                           000004F0 R        11
REM_CIR_QUAL                      00000501 R        11
REM_CIR_TYPE                      00000511 R        11
REM_CIR_TYPE_LOOP                 0000051F R        11
REM_CIR_TYPE_STRING               0000052B R        11
REM_CLUSTER                       000003BD R        11
REM_CON                           00000556 R        11
REM_CONID_ADRS                    000006A7 R        09
REM_CON_QUAL                      00000567 R        11
REM_CON_TYPE                      00000577 R        11
REM_CON_TYPE_LOOP                 00000585 R        11
REM_CON_TYPE_STRING               00000591 R        11
REM_COUNTERS                      000003D7 R        11
REM_CREDITS                       000003E4 R        11
REM_ERRORS                        000003FE R        11
REM_ID                            00000D18 RG       10
REM_ID_LOOP                       00000442 R        11
REM_LOCAL_PORTS                   000003F1 R        11
REM_MEMBERS                       000003CA R        11
REM_NEXT_CLASS                    000003A1 R        11
REM_NODE                          00000CFC RG       10
REM_NODE_LOOP                     000004D6 R        11
REM_PROC_NAME_ADRS                0000063D R        09
REM_STATE_ADRS                    00000563 R        09
REM_SYS                           0000040B R        11
REM_SYS_ID                        00000434 R        11
REM_SYS_ID_STRING                 0000044E R        11
REM_SYS_NODE                      000004C8 R        11
REM_SYS_NODE_STRING               000004E2 R        11
REM_SYS_QUAL                      0000041C R        11
REM_SYS_TYPE                      0000045C R        11
REM_SYS_TYPE_LOOP                 0000046A R        11
REM_SYS_TYPE_STRING               00000476 R        11
REPORT_QUAL                       0000001B R        02
RESET                             00001129 RG       10
RET_BUF                           00001DB3 R        03
RET_BUF_DSCR                      00001DAB R        03
RET_BUF_LEN                     = 0000003F
RIGHT_T                         = 00000075
RMS$_DEV                          ********     X    10
RMS$_EOF                          ********     X    10
RPORT_NUM_ADRS                    000004B6 R        09
RPORT_TYP_STRNG                   000002CA R        02
RPORT_TYP_STRNG_LEN             = 00000006
RPT$M_CLSTR                     = 00000001
RPT$V_CLSTR                     = 00000000
RP_FUNCTIONS_ADRS                 000005CE R        09
RP_OWNER_ADRS                     000004D8 R        09
RP_REVISION_ADRS                  000005AA R        09
RP_TYPE_ADRS                      000004F9 R        09
RSPS_QUEUED_ADRS                  00000255 R        0E
SAVED_CLASSES                     000021FC R        03
SAVE_CHAR_BUF                     00002214 R        03
SAVE_OLD                          00000835 R        0F
SAVE_TITLE_LEN                    00002252 R        0F
SB$B_HWVERS                     = 00000038
SB$B_HWVERS_DSCR                  00000148 R        09
SB$B_SYSTEMID                   = 00000018
SB$B_SYSTEMID_DSCR                00000129 R        09
SB$C_LENGTH                     = 00000060
SB$L_BLINK                      = 00000004
SB$L_CSB                        = 0000005C
SB$L_FLINK                      = 00000000
SB$L_PBFL                       = 0000000C
SB$Q_SWINCARN                   = 0000002C
```

```
SB$Q_SWINCARN_2                 = 0000002C
SB$Q_SWINCARN_2_DSCR              00000233 R       09
SB$Q_SWINCARN_DSCR                0000020F R       09
SB$SHC$L_NUM_PB                 = 00000004
SB$SHC$L_NUM_PB_DSCR              000001A9 R       09
SB$T_HWTYPE                     = 00000034
SB$T_HWTYPE_DSCR                  00000168 R       09
SB$T_NODENAME                   = 00000044
SB$T_NODENAME_DSCR                0000010C R       09
SB$T_SWTYPE                     = 00000024
SB$T_SWVERS                     = 00000028
SB$T_SWVERS_DSCR                  00000188 R       09
SB$W_MAXDG                      = 00000020
SB$W_MAXDG_DSCR                   000001CE R       09
SB$W_MAXMSG                     = 00000022
SB$W_MAXMSG_DSCR                  000001EE R       09
SB$W_SIZE                       = 00000008
SB_BUF_PNTR                       00000017 R       10
SB_COUNT                          000003E5 R       03
SB_DSCR                           00000027 R       10
SB_FIRST_LOOP                     00002E87 R       0F
SB_IN                           = 00000002
SB_LOOP                           00002E64 R       0F
SB_LOOP_END                       00003029 R       0F
SB_SNAP                           00002E40 R       0F
SCAN_5                          = 00000071
SCR$M_MORE                      = 00000010
SCR$M_MORE_DWN                  = 00000008
SCR$M_MORE_LFT                  = 00000002
SCR$M_MORE_RT                   = 00000001
SCR$M_MORE_UP                   = 00000004
SCR$V_MORE                      = 00000004
SCR$V_MORE_DWN                  = 00000003
SCR$V_MORE_LFT                  = 00000001
SCR$V_MORE_RT                   = 00000000
SCR$V_MORE_UP                   = 00000002
SCRATCH                           00000018 R       03
SCREEN_BUF                        00000563 R       03
SCREEN_BUF_LEN                  = 00000200
SCREEN_DSCR                       0000055B R       03
SCREEN_WIDTH                      00002222 R       03
SCROLL_DELAY                      00000458 R       03
SCROLL_DIF                        00000444 R       03
SCROLL_FLAGS                      0000047A R       03
SCROLL_LINES                      00000430 R       03
SCROLL_TWO                        00001B5F R       0F
SCS$B_CON_DAT                   = 00000024
SCS$GA_LOCALSB                    ********     X   0F
SCS$GB_SYSTEMID                   ********     X   0F
SCS$GL_BDT                        ********     X   0F
SCS$GL_CDL                        ********     X   0F
SCS$GQ_CONFIG                     ********     X   0F
SCS$GQ_DIRECT                     ********     X   0F
SCS$GW_BDTCNT                     ********     X   0F
SCS$GW_CDTCNT                     ********     X   0F
SCS$T_DST_PROC                  = 00000004
SCS$T_DST_PROC_DSCR               000005FF R       09
SCS$T_SRC_PROC                  = 00000014
SCS$T_SRC_PROC_DSCR               00000625 R       09
SCS_DIR_LOOP                      0000305A R       0F
SCS_HEADER_SIZE                   000021CC R       03
SCS_NUM_FIELDS                    00000426 R       03
SCS_REPORT                        000012B5 R       0F
SCS_STATE_ADRS                    000006C9 R       09
SCS_WAITERS_ADRS                  000005F3 R       09
SCS_WAITER_LOOP                   00002F42 R       0F
SCS_WAITER_SNAP                   00002F37 R       0F
SCS_WT_LP_END                     00002F4E R       0F
SDIR$B_PROCNAM                  = 0000000C
SDIR$L_CONID                    = 0000002C
SECOND_CHANCE                     0000040F RG      0F
SEND_ADRS                         0000087B R       09
SEND_SQ_ADRS                      0000028A R       09
SET                               000005BC R       11
SET_AST                           00001A39 R       0F
SET_CHAN                          00002238 R       03
SET_FIELD                         000005D0 R       11
SET_FIELD_FORMAT                  000005FC R       11
SET_FIELD_QUAL                    000005E0 R       11
SET_FIELD_WIDTH                   000005F2 R       11
SET_FORMAT                        00000F34 RG      10
SET_INTERVAL                      00000F98 RG      10
SET_NEXT                          00000618 R       11
SET_REGION_COMMAND                000019EA R       0F
SET_REGION_DATA                   000019DC R       0F
SET_REGION_FULL                   000019CE R       0F
SET_REGION_HELP                   00001A03 R       0F
SET_SCREEN                        00000FBA RG      10
SET_SCROLL                        000016B9 R       0F
SET_WIDTH                         00000F03 RG      10
SHC$C_CDT_LENGTH                = 0000008C
SHC$L_CDT_BEGIN                 = 00000014
SHC$L_CDT_DIF                   = FFFFFFF0
SHC$L_PDT_BEGIN                 = 000000B4
SHC$L_PDT_DIF                   = FFFFFF4C
SHC$L_PDT_LENGTH                = 000000D0
SHC$SB$B_DIS_POS                = 00000009
SHOW                              00000030 R       11
SHOW$_AMBIG                       ********     X   0F
SHOW$_CACLS                       ********     X   10
SHOW$_CRCLS                       ********     X   10
SHOW$_HIBER                       ********     X   0F
SHOW$_IVADDID                     ********     X   10
SHOW$_IVADDNODE                   ********     X   10
SHOW$_IVREMID                     ********     X   10
SHOW$_IVREMNODE                   ********     X   10
SHOW$_NOAVAIL                     ********     X   10
SHOW$_UNRCMD                      ********     X   0F
SHOW$_UNRKEY                      ********     X   0F
SHOW_CLUSTER                      00000929 RG      10
SHOW_CLUSTER$COMMAND_TABLE        ********     X   0F
SHOW_LOCAL                        0000099F RG      10
SIGNAL                            0000155D R       0F
SNAPSHOT                          00002DE6 RG      0F
```

```
SNAPSHOT_END                 000032D1 R      0F       TIME_DSCR                      000020B5 R      03
SNAPSHOT_ERR                 000032D4 R      0F       TIME_LEN                     = 00000014
SNAP_ADRS                    0000056A R      02       TIME_SOURCE                    00000000 R      02
SNAP_BUF                     00000000 R      10       TIME_STRING_LEN              = 0000000F
SNAP_BUF_END                 000004A3 R      10       TITLE_BUF                      00001193 R      03
SNAP_BUF_LEN               = 000004A3                 TITLE_BUF_LEN                = 00000400
SNAP_END                     00003333 R      0F       TITLE_COUNT                    000021BC R      03
SOFTWARE_ADRS                000001A0 R      09       TITLE_DSCR                     0000118B R      03
SS$_BADPARAM               = 00000014                 TITLE_LEN_ARRAY                000021D0 R      03
SS$_INTERLOCK              = 0000038C                 TITLE_START                    000021C0 R      03
SS$_NOCMKRNL               = 00002804                 TKN_DSCR                       00000495 R      03
SS$_NORMAL                 = 00000001                 TOP_CMD_LINE                   00000470 R      03
SS$_NOSUCHNODE             = 0000028C                 TOP_T                        = 00000077
SS$_NOTRAN                 = 00000629                 TPA$K_COUNT0                 = 00000008
START                        00000000 RG     0F       TPA$K_LENGTH0                = 00000024
START_BIT                    00002270 R      03       TPA$L_PARAM                  = 00000020
STATUS                       00000491 R      03       TPA$L_STRINGCNT              = 00000008
STATUS_ADRS                  0000044C R      09       TPA$L_STRINGPTR              = 0000000C
STORE_MSG                    000004E9 R      0F       TPA$L_TOKENCNT               = 00000010
STR$UPCASE                   ********   X    0F       TPA$L_TOKENPTR               = 00000014
SWITCH                       000021AB R      03       TPA$M_ABBREV                 = 00000002
SWITCH_DISPLAY               00002566 R      0F       TPA$_ALPHA                   = 000001EE
SYM_PRFX                     00000489 R      02       TPA$_ANY                     = 000001ED
SYS$ASCTIM                   ********  GX    0F       TPA$_BLANK                   = 000001F2
SYS$ASSIGN                   ********  GX    0F       TPA$_DECIMAL                 = 000001F3
SYS$CANTIM                   ********  GX    0F       TPA$_DIGIT                   = 000001EF
SYS$CLOSE                    ********  GX    10       TPA$_EOS                     = 000001F7
SYS$CLREF                    ********  GX    0F       TPA$_EXIT                    = FFFFFFFF
SYS$CMKRNL                   ********  GX    0F       TPA$_FAIL                    = FFFFFFFE
SYS$CONNECT                  ********  GX    10       TPA$_FILESPEC                = 000001EA
SYS$DCLEXH                   ********  GX    0F       TPA$_HEX                     = 000001F5
SYS$EXIT                     ********  GX    0F       TPA$_IDENT                   = 000001EC
SYS$GET                      ********  GX    10       TPA$_KEYWORD                 = 00000100
SYS$GETDVI                   ********  GX    0F       TPA$_LAMBDA                  = 000001F6
SYS$GETJPI                   ********  GX    0F       TPA$_MAXKEY                  = 000000DC
SYS$GETMSG                   ********  GX    0F       TPA$_OCTAL                   = 000001F4
SYS$HIBER                    ********  GX    0F       TPA$_STRING                  = 000001F0
SYS$LKWSET                   ********  GX    0F       TPA$_SUBXPR                  = 000001F8
SYS$OPEN                     ********  GX    10       TPA$_SYMBOL                  = 000001F1
SYS$PARSE                    ********  GX    0F       TPA$_UIC                     = 000001EB
SYS$QIO                      ********  GX    0F       TPARSE_BLK                     000004C7 R      03
SYS$QIOW                     ********  GX    0F       TRANSITION_TIME_ADRS           0000046B R      09
SYS$READEF                   ********  GX    0F       TRAN_BUF                       000003A4 R      03
SYS$SCHDWK                   ********  GX    0F       TRAN_DSCR                      0000039C R      03
SYS$SETEF                    ********  GX    0F       TRM$M_TM_NOEDIT              = 00008000
SYS$SETIMR                   ********  GX    0F       TRM$_MODIFIERS               = 00000000
SYS$TRNLOG                   ********  GX    0F       TRY_LOCK                       00003314 R      0F
SYS$ULWSET                   ********  GX    0F       TT$M_ESCAPE                  = 00000008
SYS$WFLOR                    ********  GX    0F       TT$M_NOBRDCST                = 00020000
SYS_ID_ADRS                  00000141 R      09       TT$M_SCOPE                   = 00001000
SYS_ID_LIST                  00002278 R      03       TT$M_WRAP                    = 00000200
TERM_LIST                    00000572 R      02       TT2$M_AVO                    = 08000000
TERM_LIST_LEN              = 0000000C                 TT2$M_BRDCSTMBX              = 00000010
TIMBUF                       00001F96 R      03       TT2$M_DECCRT                 = 20000000
TIME                       = 00000005                 TYPEAHD_BUF                    000004BD R      03
TIME_BUF                     000020BD R      03       UCB$B_DEVTYPE                = 00000041
TIME_DISPLAY                 0000259B R      0F       UCB$B_ERTCNT                 = 00000080
```

```
UCB$L_PDT                        = 00000084
UCB$V_ONLINE                     = 00000004
UCB$W_STS                        = 00000064
UL_CORNER                        = 0000006C
UNACKED_ADRS                       00000309 R     09
UR_CORNER                        = 0000006B
VALUE                              0000054F R     03
VAR_BUF                            000003ED R     03
VAR_BUF_ADRS                       00000047 R     10
VAR_BUF_END                        000003F1 R     03
VAR_BUF_LEN                        000003F5 R     03
VERT_BAR                         = 00000078
VOTES_ADRS                         000003AD R     09
VT52_DWN_ARROW                   = 00000042
VT52_LFT_ARROW                   = 00000043
VT52_RT_ARROW                    = 00000044
VT52_UP_ARROW                    = 00000041
V_SCROLL                           0000045C R     03
V_SCROLL_LINE                      00000460 R     03
WARMCDRPS_ADRS                     00000369 R     09
WAS_LOCKED                         0000330A R     0F
WIDE                             = 00000084
YES_NO_STRING                      00000404 R     02
YES_NO_STRNG_LEN                 = 00000003
```

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

PSECT name              Allocation          PSECT No.  Attributes
----------              ----------          ---------  ----------
.  ABS  .               00000000  (    0.)  00 (  0.)  NOPIC  USR  CON  ABS  LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                   00000360  (  864.)  01 (  1.)  NOPIC  USR  CON  ABS  LCL NOSHR   EXE   RD    WRT NOVEC BYTE
RODATA                  000005CE  ( 1486.)  02 (  2.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD  NOWRT NOVEC BYTE
RWDATA                  000022E4  ( 8932.)  03 (  3.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
$RMSNAM                 00000010  (   16.)  04 (  4.)  NOPIC  USR  CON  REL  LCL NOSHR   EXE   RD    WRT NOVEC BYTE
CLUSTER_SIZE_ARRAY      00000046  (   70.)  05 (  5.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
CLUSTER_BITMAP          00000018  (   24.)  06 (  6.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
CLUSTER_TABLE           00000118  (  280.)  07 (  7.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD  NOWRT NOVEC BYTE
CLUSTER_KEYTABLE        00000234  (  564.)  08 (  8.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD  NOWRT NOVEC BYTE
CLUSTER_DSCRS           00000923  ( 2339.)  09 (  9.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
LOCAL_SIZE_ARRAY        00000019  (   25.)  0A ( 10.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
LOCAL_BITMAP            00000008  (    8.)  0B ( 11.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
LOCAL_TABLE             00000064  (  100.)  0C ( 12.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD  NOWRT NOVEC BYTE
LOCAL_KEYTABLE          000000CC  (  204.)  0D ( 13.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD  NOWRT NOVEC BYTE
LOCAL_DSCRS             00000369  (  873.)  0E ( 14.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
MAIN                    00003333  (13107.)  0F ( 15.)  NOPIC  USR  CON  REL  LCL NOSHR   EXE   RD  NOWRT NOVEC BYTE
SNAP_DATA               00001362  ( 4962.)  10 ( 16.)  NOPIC  USR  CON  REL  LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
_LIB$STATE$             00000652  ( 1618.)  11 ( 17.)    PIC  USR  CON  REL  LCL   SHR   EXE   RD  NOWRT NOVEC BYTE
_LIB$KEY0$              00000098  (  152.)  12 ( 18.)    PIC  USR  CON  REL  LCL   SHR   EXE   RD  NOWRT NOVEC WORD
_LIB$KEY1$              000001F5  (  501.)  13 ( 19.)    PIC  USR  CON  REL  LCL   SHR   EXE   RD  NOWRT NOVEC WORD
```

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

Phase                       Page faults     CPU Time        Elapsed Time
-----                       -----------     --------        ------------
Initialization                       10     00:00:00.07     00:00:00.57
Command processing                   74     00:00:00.92     00:00:05.42
Pass 1                             2671     00:02:27.22     00:07:54.26
Symbol table sort                     0     00:00:07.28     00:00:27.10
Pass 2                             1416     00:00:43.32     00:02:28.14
Symbol table output                   5     00:00:01.04     00:00:03.65
Psect synopsis output                 1     00:00:00.07     00:00:00.14
Cross-reference output                0     00:00:00.00     00:00:00.00
Assembler run totals               4180     00:03:19.92     00:10:59.28

The working set limit was 4100 pages.
851980 bytes (1665 pages) of virtual memory were used to buffer the intermediate code.
There were 240 pages of symbol table space allocated to hold 4034 non-local and 603 local symbols.
13210 source lines were read in Pass 1, producing 224 object records in Pass 2.
134 pages of virtual memory were used to define 114 macros.

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SHRLIB]PALIB.MLB;1                         2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         17
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      60
TOTALS (all libraries)                                 79

3970 GETS were required to define 79 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SHWCLSTR/OBJ=OBJ$:SHWCLSTR MSRC$:SHWCLSTR/UPDATE=(ENH$:SHWCLSTR)+EXECML$/LIB+SHRLIB$:PALIB/LIB
```

0058 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SHWCLSTR
LIS

SHOWUSER
LIS

0059 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TERMDEFS
LIS

SUBMITMSG
LIS

TRANQUEUE
LIS

SWITCHTRM
LIS

SUBMIT.
LIS