K 6

```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

FILEID**ERFPARSER

```
EEEEEEEEEE  RRRRRRRR    FFFFFFFFFF  PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE  RRRRRRRR
EEEEEEEEEE  RRRRRRRRR   FFFFFFFFFF  PPPPPPPPP     AAAAAA    RRRRRRRRR     SSSSSSSS  EEEEEEEEEE  RRRRRRRRR
EE          RR      RR  FF          PP      PP   AA      AA  RR      RR  SS        SS  EE          RR      RR
EE          RR      RR  FF          PP      PP   AA      AA  RR      RR  SS            EE          RR      RR
EE          RR      RR  FF          PP      PP   AA      AA  RR      RR  SS            EE          RR      RR
EE          RR      RR  FF          PP      PP   AA      AA  RR      RR  SS            EE          RR      RR
EEEEEEEE    RRRRRRRRR   FFFFFFF     PPPPPPPPP    AA      AA  RRRRRRRRR     SSSSSS    EEEEEEEE    RRRRRRRRR
EEEEEEEE    RRRRRRRRR   FFFFFFF     PPPPPPPP     AA      AA  RRRRRRRRR       SSSSSS  EEEEEEEE    RRRRRRRRR
EE          RR   RR     FF          PP           AAAAAAAAAA  RR   RR             SS  EE          RR   RR
EE          RR   RR     FF          PP           AAAAAAAAAA  RR   RR             SS  EE          RR   RR
EE          RR    RR    FF          PP           AA      AA  RR    RR            SS  EE          RR    RR
EE          RR    RR    FF          PP           AA      AA  RR    RR            SS  EE          RR    RR
EEEEEEEEEE  RR      RR  FF          PP           AA      AA  RR      RR  SSSSSSSS  EEEEEEEEEE  RR      RR
EEEEEEEEEE  RR      RR  FF          PP           AA      AA  RR      RR  SSSSSSSS  EEEEEEEEEE  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
```

```
;  1          0001  0 MODULE ERFPARSER
;  2          0002  0 (%TITLE 'Command Parser'
;  3          0003  0 IDENT = 'V04-000') =
;  4          0004  0
;  5          0005  1 BEGIN
;  6          0006  1
;  7          0007  1 !
;  8          0008  1 !*****************************************************************************
;  9          0009  1 !*                                                                           *
; 10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
; 11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
; 12          0012  1 !*  ALL RIGHTS RESERVED.                                                    *
; 13          0013  1 !*                                                                         *
; 14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
; 15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 19          0019  1 !*  TRANSFERRED.                                                            *
; 20          0020  1 !*                                                                         *
; 21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 23          0023  1 !*  CORPORATION.                                                            *
; 24          0024  1 !*                                                                         *
; 25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
; 27          0027  1 !*                                                                         *
; 28          0028  1 !*                                                                         *
; 29          0029  1 !*****************************************************************************
; 30          0030  1
; 31          0031  1 !++
; 32          0032  1 ! FACILITY:  ERF, Error Log Report Generator
; 33          0033  1 !
; 34          0034  1 ! ABSTRACT:
; 35          0035  1 !
; 36          0036  1 !       This module contains the routines that perform the command parsing
; 37          0037  1 !       for ERF.
; 38          0038  1 !
; 39          0039  1 ! ENVIRONMENT:
; 40          0040  1 !
; 41          0041  1 !       VAX/VMS operating system, user mode.
; 42          0042  1 !
; 43          0043  1 ! AUTHOR:  Sharon Reynolds,     CREATION DATE:  October 1982
; 44          0044  1 !
; 45          0045  1 ! Modified by:
; 46          0046  1 !
; 47          0047  1 !       V03-013 SAR0273         Sharon A. Reynolds      18-Jun-1984
; 48          0048  1 !               - Fixed a bug with the parsing of device names.
; 49          0049  1 !
; 50          0050  1 !       V03-012 SAR0266         Sharon A. Reynolds      15-May-1984
; 51          0051  1 !               - Re-inserted code for handling /inc=star$, lost due to
; 52          0052  1 !               incorrect version checked in.
; 53          0053  1 !
; 54          0054  1 !       V03-011 SAR0255         Sharon A. Reynolds      23-Apr-1984
; 55          0055  1 !               - Fixed a problem in parsing nodes names of max length.
; 56          0056  1 !               - Added a check for /output and /binary.
; 57          0057  1 !
```

```
;    58          0058  1 !        V03-010 SAR0242         Sharon A. Reynolds      4-Apr-1984
;    59          0059  1 !                - Removed unnecessary code from search_queue.
;    60          0060  1 !                - Made unit number max = 5.
;    61          0061  1 !
;    62          0062  1 !        V03-009 EAD0120         Elliott A. Drayton              23-Feb-1984
;    63          0063  1 !                Changed code to handel UNKNOWN as a keyword, not as a qualifier.
;    64          0064  1 !
;    65          0065  1 !        V03-008 SAR0190         Sharon A. Reynolds,     13-Feb-1984
;    66          0066  1 !                - Added additional test for summary updates.
;    67          0067  1 !                - Added 'CS' device name support to the device table
;    68          0068  1 !                search routine.
;    69          0069  1 !                - Changed the error return for lib$cvt_xxx.
;    70          0070  1 !
;    71          0071  1 !                JMG0008         Joel M. Gringorten,      6-Feb-1984
;    72          0072  1 !                Added Statistics qualifier support.
;    73          0073  1 !
;    74          0074  1 !        V03-007 JMG0001         Joel M. Gringorten,      9-Jan-1984
;    75          0075  1 !                Added support for SUMMARY=HISTOGRAM.
;    76          0076  1 !
;    77          0077  1 !        V03-006 SAR0180         Sharon A. Reynolds,     13-Dec-1983
;    78          0078  1 !                - Removed unnecessary descriptors.
;    79          0079  1 !                - Added unsolicited_mscp keyword.
;    80          0080  1 !                - Removed the logmessage keyword.
;    81          0081  1 !                - Added the attentions keyword.
;    82          0082  1 !                - Fixed the parsing of 'sloth$dba3'.
;    83          0083  1 !                - Changed the name wild indicator to node name wild indicator.
;    84          0084  1 !                - Made the 'parse_devname' and 'search_queue' routines
;    85          0085  1 !                  support node name wild indicator.
;    86          0086  1 !
;    87          0087  1 !        V03-005 SAR0166         Sharon A. Reynolds,     14-Oct-1983
;    88          0088  1 !                - Made the following command valid. (/inclu=disk/excl=db).
;    89          0089  1 !                - Changed the way the report type is referenced.
;    90          0090  1 !                - Removed reference to erf_norep.
;    91          0091  1 !
;    92          0092  1 !        V03-004 SAR0151         Sharon A. Reynolds,     7-Oct-1983
;    93          0093  1 !                Fixed a bug in GET_DEVICE_SELECT.
;    94          0094  1 !
;    95          0095  1 !        V03-003 SAR0121         Sharon A. Reynolds,     23-Aug-1983
;    96          0096  1 !                Fixed a problem with report type selection (/NOFULL)
;    97          0097  1 !                and added /REJECTED qualifier support. Re-wrote the
;    98          0098  1 !                search routine for use with the permanent device tables.
;    99          0099  1 !
;   100          0100  1 !        V03-002 SAR0028         Sharon A. Reynolds,     11-May-1983
;   101          0101  1 !                Removed support for logstatus keyword. Fixed a
;   102          0102  1 !                problem in parsing an '*' in a device name spec.
;   103          0103  1 !
;   104          0104  1 !        V03-001 SAR0014         Sharon A. Reynolds,     18-Apr-1983
;   105          0105  1 !                Fixed a problem with error message returns for the parsing
;   106          0106  1 !                of /include and /exclude device name and keyword selection.
;   107          0107  1 !
;   108          0108  1 !--
;   109          0109  1 !
;   110          0110  1
;   111          0111  1 !
;   112          0112  1 ! This global data psect is quadword aligned. It currently
;   113          0113  1 ! contains data for use by the LIB$INSQTI routine.
;   114          0114  1 !
```

```
;  115          0115  1 PSECT GLOBAL = QUEUE_DATA (PIC,ALIGN (3)) ;
;  116          0116  1 Global
;  117          0117  1     Root_flink:         Initial (0),
;  118          0118  1     Root_blink:         Initial (0),
;  119          0119  1     Que_addrs:          Initial (0),
;  120          0120  1     Que_entry_addrs:    Initial (0) ;
;  121          0121  1
;  122          0122  1 PSECT
;  123          0123  1     Code = $CODE (pic,addressing_mode(general)),
;  124          0124  1     Plit = $PLIT (pic,addressing_mode(general)),
;  125          0125  1     Own = $own$ (pic,addressing_mode(general)),
;  126          0126  1     Global = $global$ (pic,addressing_mode(general)) ;
;  127          0127  1
;  128          0128  1
;  129          0129  1
;  130          0130  1 !
;  131          0131  1 ! Required files
;  132          0132  1 !
;  133          0133  1 REQUIRE 'SRC$:ERFDEF.REQ' ;              ! For message definitions
;  134          0419  1 REQUIRE 'LIB$:PARSERDAT.R32' ;          ! ERF parser data definitions
;  135          0573  1
;  136          0574  1 !
;  137          0575  1 ! Table of contents
;  138          0576  1 !
;  139          0577  1 FORWARD ROUTINE
;  140          0578  1     Class_option_check: NOVALUE,
;  141          0579  1     Device_option_check,
;  142          0580  1     Get_device_select,
;  143          0581  1     Get_vm,
;  144          0582  1     Parse_command,
;  145          0583  1     Parse_devname,
;  146          0584  1     Search_queue,
;  147          0585  1     Translate_device ;
;  148          0586  1
;  149          0587  1 !
;  150          0588  1 ! Declare external routines
;  151          0589  1 !
;  152          0590  1 EXTERNAL ROUTINE
;  153          0591  1     CLI$GET_VALUE: addressing_mode(general),    ! Get parameter or qualifier
;  154          0592  1                                                 ! value.
;  155          0593  1     CLI$PRESENT: addressing_mode (general),     ! Determine if entity is present
;  156          0594  1     LIB$LOOKUP_KEY : addressing_mode (general), ! Match selected keyword against
;  157          0595  1                                                 ! the specified keyword table.
;  158          0596  1     LIB$CVT_TIME: addressing_mode (general),    ! Convert time string to binary
;  159          0597  1                                                 ! value.
;  160          0598  1     LIB$CVT_DTB: addressing_mode (general),     ! Convert decimal to binary
;  161          0599  1     LIB$CVT_HTB: addressing_mode (general),     ! Convert hexadecimal to binary
;  162          0600  1     LIB$GET_VM: addressing_mode (general),      ! Get virtual memory
;  163          0601  1     LIB$INSQTI: addressing_mode (general),      ! Insert entry at head of queue
;  164          0602  1     LIB$REMQTI: addressing_mode (general) ;     ! Remove an entry from the
;  165          0603  1                                                 ! head for the queue.
;  166          0604  1
;  167          0605  1 !
;  168          0606  1 ! Declare external literals
;  169          0607  1 !
;  170          0608  1 EXTERNAL LITERAL
;  171          0609  1     Cli$_absent,
```

```
;  172     0610  1      Cli$_negated,
;  173     0611  1      Cli$_present,
;  174     0612  1      Erf_cnfquaval,
;  175     0613  1      Erf_cvterr,
;  176     0614  1      Erf_devselreq ;
;  177     0615  1
;  178     0616  1  !
;  179     0617  1  ! Declare literals
;  180     0618  1  !
;  181     0619  1  GLOBAL LITERAL
;  182     0620  1  !    Async = 0,                  ! Index for include_class/mask and
;  183     0621  1      Bus = 1,                    ! exclude_class/mask structures
;  184     0622  1      Disk = 2,
;  185     0623  1      Realtime = 3,
;  186     0624  1      Sync = 4,
;  187     0625  1      Tape = 5,
;  188     0626  1
;  189     0627  1      Error = 2,                  !
;  190     0628  1      Max_class = (5),            ! Maximum number of device class selections for /include,/exclude
;  191     0629  1      Q_entry_size = (((dev$s_dev_queue+7)/8)*8) ;! Device queue entry size
;  192     0630  1
;  193     0631  1  !
;  194     0632  1  ! Declare global storage
;  195     0633  1  !
;  196     0634  1  GLOBAL
;  197     0635  1  !    Bugchk_type,                          ! Storage for bugcheck values
;  198     0636  1      Dev_class_key,                        ! Device class selection indicator
;  199     0637  1      Dev_entry_key,                        ! Device entry selection indicator
;  200     0638  1      Dev_name,                             ! Device name (first two chars)
;  201     0639  1      Dev_select:          REF $BBLOCK,     ! Device selection que entry storage
;  202     0640  1      Entry_value,                          ! Converted entry value storage
;  203     0641  1      Exclude_class:       VECTOR [6,BYTE],! Selected device classes for /exclude
;  204     0642  1      Exclude_flag:        BYTE,           ! /include, /exclude processing indicator
;  205     0643  1      Exclude_key:         VECTOR [6,BYTE],!
;  206     0644  1      Exclude_mask,                        ! Exclude selection indicators
;  207     0645  1      Exclude_q_entry_cnt: BYTE,
;  208     0646  1      Include_q_entry_cnt: BYTE,
;  209     0647  1      Include_class:       VECTOR [6,BYTE],! Selected device classes for /include
;  210     0648  1      Include_key:         VECTOR [6,BYTE],!
;  211     0649  1      Include_mask,                        ! Include selection indicators
;  212     0650  1      Option_flag,                         ! Option selection indicators
;  213     0651  1      Parser_data,                         ! Address of actual data storage area
;  214     0652  1      Parser_table,                        ! Address of descriptor storage area
;  215     0653  1      Que_entry_cnt:       WORD,           ! Number of entries in the queue
;  216     0654  1      Summary_flag,                        ! Summary option selections
;  217     0655  1      Class_dir,
;  218     0656  1      Wild_carded_device ;                 ! Device wild carded
;  219     0657  1
;  220     0658  1  OWN
;  221     0659  1      Keywrd_mask:         Initial (%X'FFFFF') ;
;  222     0660  1
;  223     0661  1  MAP
;  224     0662  1      Exclude_mask:        REF $BBLOCK,
;  225     0663  1      Include_mask:        REF $BBLOCK,
;  226     0664  1      Option_flag:         REF $BBLOCK,
;  227     0665  1      Parser_data:         REF $BBLOCK,
;  228     0666  1      Parser_table:        REF $BBLOCK,
```

```
;  229          0667  1     Summary_flag:       REF $BBLOCK,
;  230          0668  1     Que_entry_addrs:    REF $BBLOCK,
;  231          0669  1     Class_dir:          REF $BBLOCK ;
;  232          0670  1
;  233          0671  1
;  234          0672  1 !
;  235          0673  1 ! Macro definitions
;  236          0674  1 !
;  237          0675  1 MACRO
;  238          0676  1     !
;  239          0677  1     ! Define a macro for converting the entry values from ascii to a value.
;  240          0678  1     !
;  241        M 0679  1     CVT_ENTRY_VALUE =
;  242        M 0680  1         Begin
;  243        M 0681  1
;  244        M 0682  1         If NOT (status = LIB$CVT_DTB (.wrk_desc[dsc$w_length],
;  245        M 0683  1                                       .wrk_desc[dsc$a_pointer],entry_value))
;  246        M 0684  1         Then
;  247        M 0685  1             !
;  248        M 0686  1             ! Error converting the ascii decimal value to a value, notify
;  249        M 0687  1             ! the user and exit.
;  250        M 0688  1             !
;  251        M 0689  1             Signal (erf_cvterr, 2,.wrk_desc[dsc$w_length],
;  252        M 0690  1                                   .wrk_desc[dsc$a_pointer]) ;
;  253          0691  1         End % ;
;  254          0692  1
```

```
;  256          0693  1 GLOBAL ROUTINE PARSE_COMMAND =                    ! Command line parsing
;  257          0694  1
;  258          0695  1 !++
;  259          0696  1 !
;  260          0697  1 ! Functional Description:
;  261          0698  1 !
;  262          0699  1 !    This routine is called from the main loop to parse the
;  263          0700  1 !    command line
;  264          0701  1 !
;  265          0702  1 ! Calling Sequence:
;  266          0703  1 !
;  267          0704  1 !    PARSE_COMMAND ()
;  268          0705  1 !
;  269          0706  1 ! Input Parameters:
;  270          0707  1 !
;  271          0708  1 !    None
;  272          0709  1 !
;  273          0710  1 ! Output Parameters:
;  274          0711  1 !
;  275          0712  1 !    *
;  276          0713  1 !
;  277          0714  1 !--
;  278          0715  2 Begin
;  279          0716  2
;  280          0717  2 !
;  281          0718  2 ! Generate static string descriptors for the qualifiers.
;  282          0719  2 !
;  283        P 0720  2 SD( '$LINE',
;  284        P 0721  2     'FILE_SPECS',
;  285        P 0722  2     'BEFORE',
;  286        P 0723  2     'BINARY',
;  287        P 0724  2     'BRIEF',
;  288        P 0725  2     'ENTRY',
;  289        P 0726  2     'EXCLUDE',
;  290        P 0727  2     'FULL',
;  291        P 0728  2     'INCLUDE',
;  292        P 0729  2     'LOG',
;  293        P 0730  2     'OUTPUT',
;  294        P 0731  2     'PAGE',
;  295        P 0732  2     'REGISTER_DUMP',
;  296        P 0733  2     'REJECTED',
;  297        P 0734  2     'SID_REGISTER',
;  298        P 0735  2     'SINCE',
;  299        P 0736  2     'STATISTICS',
;  300          0737  2     'SUMMARY') ;
;  301          0738  2
;  302          0739  2     Bind %NAME('ENTRY_END_DESC') = $DESCRIPTOR('ENTRY.END') ;
;  303          0740  2     Bind %NAME('ENTRY_START_DESC') = $DESCRIPTOR('ENTRY.START') ;
;  304          0741  2     Bind ptr_0 = CH$PTR (UPLIT ('0')) ;
;  305          0742  2     Bind ptr_z = CH$PTR (UPLIT ('z')) ;
;  306          0743  2
;  307          0744  2 LOCAL
;  308          0745  2     Cmd_line_desc,                          ! Command line desc address storage
;  309          0746  2     Full_negate:        BYTE,               ! /nofull indicator
;  310          0747  2     I:                  Initial (0),        ! Multiple uses
;  311          0748  2     Key_value,                              ! Key word value storage
;  312          0749  2     Status,                                 ! Return status storage
```

```
;   313          0750  2     System_id:          Initial (0) ;   ! Temporary storage for system id
;   314          0751  2
;   315          0752  2 OWN
;   316          0753  2     Wrk_desc:   $BBLOCK [dsc$k_d_bln]   ! Dynamic work descriptor
;   317          0754  2                 Preset ([dsc$b_class] = dsc$k_class_d),
;   318          0755  2
;   319          0756  2 !
;   320          0757  2 ! Create the keyword tables for the LIB$LOOKUP_KEY routine. It (LIB$LOOKUP_KEY)
;   321          0758  2 ! will locate a matching key and return the value associated with it.
;   322          0759  2 !
;   323          0760  2 Summary_keywords:                        ! /SUMMARY keywords
;   324        P 0761  2     $LIB_KEY_TABLE (
;   325        P 0762  2         (Device, 01),
;   326        P 0763  2         (Entry, 02),
;   327        P 0764  2         (Memory, 03),
;   328        P 0765  2         (Volume, 04),
;   329          0766  2         (Histogram, 05)) ;
;   330          0767  2
;   331          0768  2 !
;   332          0769  2 ! Allocate memory for the parser table (descriptors) and the parser
;   333          0770  2 ! data (actual data).
;   334          0771  2 !
;   335          0772  2 Parser_table = GET_VM (erl$s_prs_table) ;
;   336          0773  2 Parser_data = GET_VM (erl$s_prs_data) ;
;   337          0774  2
;   338          0775  2 !
;   339          0776  2 ! Set up some default values.
;   340          0777  2 !
;   341          0778  2 Parser_data[erl$b_rpt_type] = full_rep ;        ! Default to full
;   342          0779  2 Parser_data[erl$q_end_date]+0 = (%x'FFFFFFFF') ; ! Default to most future date
;   343          0780  2 Parser_data[erl$q_end_date]+4 = (%x'7FFFFFFF') ; !
;   344          0781  2 Parser_data[erl$q_start_date]+0 = 0 ;          ! Default to earliest possible
;   345          0782  2                                              ! date/time.
;   346          0783  2 Parser_data[erl$q_start_date]+4 = 0 ;         !
;   347          0784  2 Parser_data[erl$l_start_entry] = 0 ;         ! Default to the beginning entry
;   348          0785  2 Parser_data[erl$l_end_entry] = (%x'FFFFFFFF') ; ! Default to the last entry
;   349          0786  2
;   350          0787  2 !
;   351          0788  2 ! Get virtual memory for the bugcheck value storage.
;   352          0789  2 !
;   353          0790  2 !Bugchk_type = GET_VM (bug$s_bugchk_flags) ;
;   354          0791  2
;   355          0792  2 !
;   356          0793  2 ! Get virtual memory for the exclude mask flags.
;   357          0794  2 Exclude_mask = GET_VM (exc$s_exclude_flags) ;
;   358          0795  2
;   359          0796  2 !
;   360          0797  2 ! Get virtual memory for the include mask flags.
;   361          0798  2 Include_mask = GET_VM (inc$s_include_flags) ;
;   362          0799  2
;   363          0800  2 !
;   364          0801  2 ! Get virtual memory for the option flags and set up
;   365          0802  2 ! some defaults.
;   366          0803  2 Option_flag = GET_VM (opt$s_opt_flags) ;
;   367          0804  2 Option_flag[opt$v_output_qual] = true ;
;   368          0805  2 Option_flag[opt$v_full_qual] = true ;
;   369          0806  2
```

```
;  370          0807  2 !
;  371          0808  2 ! Get virtual memory for the summary qualifier flags.
;  372          0809  2 Summary_flag = GET_VM (sum$s_summary_flags) ;
;  373          0810  2
;  374          0811  2 ! Set up the descriptor that points to the entire command line.
;  375          0812  2 !
;  376          0813  2 Parser_table[cmd$b_class] = dsc$k_class_d ;
;  377          0814  2 Cmd_line_desc = parser_table[erl$r_cmd_line] ;
;  378          0815  2
```

```
;  380     0816  2 !
;  381     0817  2 ! Save the entire command line for output at the end
;  382     0818  2 ! of any report.
;  383     0819  2 !
;  384     0820  2 CLI$GET_VALUE ($LINE_DESC,.cmd_line_desc) ;
;  385     0821  2
;  386     0822  2 !
;  387     0823  2 ! Parse the command line.
;  388     0824  2 !
;  389     0825  2 ! Determine if the /BEFORE qualifier was specified.
;  390     0826  2 !
;  391     0827  2 If CLI$PRESENT (before_desc)
;  392     0828  2 Then
;  393     0829  2     !
;  394     0830  2     ! Get any value associated with the qualifier.
;  395     0831  2     !
;  396     0832  3     Begin
;  397     0833  3     Option_flag[opt$v_before_qual] = true ;
;  398     0834  3     If CLI$GET_VALUE (before_desc,wrk_desc)
;  399     0835  3     Then
;  400     0836  3         !
;  401     0837  3         ! Convert the ascii time/date string to binary time/date.
;  402     0838  3         ! LIB$CVT_TIME handles the Today, Yesterday, and Tomorrow keywords and
;  403     0839  3         ! will convert an absolute or delta time string or a combination of
;  404     0840  3         ! the two.
;  405     0841  3         !
;  406     0842  4         Begin
;  407     0843  5         If NOT (status = LIB$CVT_TIME (wrk_desc,parser_data[erl$q_end_date]))
;  408     0844  4         Then
;  409     0845  4             !
;  410     0846  4             ! Date/time conversion error, notify the user.
;  411     0847  4             !
;  412     0848  4             Signal (.status) ;
;  413     0849  3         End ;
;  414     0850  2     End ;
;  415     0851  2
;  416     0852  2 !
;  417     0853  2 ! Determine if the  /BINARY qualifier was specified. The file spec
;  418     0854  2 ! for the /binary qualifier is retrieved and parsed by the
;  419     0855  2 ! Parse_output_files routine in ERF.
;  420     0856  2 !
;  421     0857  2 If CLI$PRESENT (binary_desc)
;  422     0858  2 Then
;  423     0859  2     !
;  424     0860  2     ! Indicate that the qualifier was specified and reset the
;  425     0861  2     ! default report type information.
;  426     0862  2     !
;  427     0863  3     Begin
;  428     0864  3     Status = CLI$PRESENT (output_desc) ;
;  429     0865  3     If .status EQL CLI$_PRESENT
;  430     0866  3     Then
;  431     0867  3         Signal_stop (msg$_confqual) ;
;  432     0868  3
;  433     0869  3     Option_flag[opt$v_binary_qual] = true ;
;  434     0870  3     Option_flag[opt$v_full_qual] = false ;
;  435     0871  3     Parser_data[erl$b_rpt_type] = 0 ;
;  436     0872  2     End ;
```

```
;  437     0873  2
;  438     0874  2 !
;  439     0875  2 ! Determine if the /ENTRY qualifier was specified. The CLD will
;  440     0876  2 ! ensure a value was specified.
;  441     0877  2 !
;  442     0878  2 If CLI$PRESENT (entry_desc)
;  443     0879  2 Then
;  444     0880  2     !
;  445     0881  2     ! Indicate that the /Entry qualifier was specified and get
;  446     0882  2     ! any associated values.
;  447     0883  2     !
;  448     0884  3     Begin
;  449     0885  3     Option_flag[opt$v_entry_qual] = true ;
;  450     0886  3
;  451     0887  3     !
;  452     0888  3     ! Get the value associated with /ENTRY=start:value. The CLD will
;  453     0889  3     ! return a default if the user did not specify anything.
;  454     0890  3     !
;  455     0891  3     If CLI$GET_VALUE(ENTRY_START_DESC,wrk_desc)
;  456     0892  3     Then
;  457     0893  3         !
;  458     0894  3         ! Convert the ascii input to a decimal value and save it. If there was
;  459     0895  3         ! a conversion error the CVT_ENTRY_VALUE will notify the user and will
;  460     0896  3         ! not return.
;  461     0897  3         !
;  462     0898  4         Begin
;  463     0899  4         CVT_ENTRY_VALUE ;
;  464     0900  4         Parser_data[erl$l_start_entry] = .entry_value ;
;  465     0901  3         End ;
;  466     0902  3     !
;  467     0903  3     ! Get the value associated with /ENTRY=end:value. The CLD will
;  468     0904  3     ! return a default if the user did not specify anything.
;  469     0905  3     !
;  470     0906  3     If CLI$GET_VALUE(ENTRY_END_DESC,wrk_desc)
;  471     0907  3     Then
;  472     0908  3         !
;  473     0909  3         ! Convert the ascii input to a decimal value and save it. If there was
;  474     0910  3         ! a conversion error the CVT_ENTRY_VALUE will notify the user and will
;  475     0911  3         ! not return.
;  476     0912  3         !
;  477     0913  4         Begin
;  478     0914  4         CVT_ENTRY_VALUE ;
;  479     0915  4         Parser_data[erl$l_end_entry] = .entry_value ;
;  480     0916  3         End ;
;  481     0917  2     End ;
;  482     0918  2
;  483     0919  2 !
;  484     0920  2 ! Determine whether the /EXCLUDE qualifier was specified.
;  485     0921  2 !
;  486     0922  2 If CLI$PRESENT (exclude_desc)
;  487     0923  2 Then
;  488     0924  2     !
;  489     0925  2     ! Indicate that the /exclude qualifier was specified.
;  490     0926  2     !
;  491     0927  3     Begin
;  492     0928  3     Option_flag[opt$v_exclude_qual] = true ;
;  493     0929  3     Exclude_flag = true ;
```

```
;  494     0930  3
;  495     0931  3      !
;  496     0932  3      ! Get any value(s) associated with the qualifier.
;  497     0933  3      !
;  498     0934  3      While CLI$GET_VALUE (exclude_desc,wrk_desc) do
;  499     0935  4          Begin
;  500     0936  4          !
;  501     0937  4          ! Determine if the retrieved 'value', is a keyword.
;  502     0938  4          !
;  503     0939  5          If NOT (GET_DEVICE_SELECT (wrk_desc))
;  504     0940  4          Then
;  505     0941  4              !
;  506     0942  4              ! Not valid input for device selection, notify the user
;  507     0943  4              ! and exit.
;  508     0944  4              !
;  509     0945  4              Signal_stop (msg$_invquaval, 2,
;  510     0946  4                           wrk_desc,
;  511     0947  4                           exclude_desc) ;
;  512     0948  3          End ;
;  513     0949  2      End ;
;  514     0950  2
;  515     0951  2
;  516     0952  2  !
;  517     0953  2  ! Determine whether the /INCLUDE qualifier was specified, get any
;  518     0954  2  ! qualifier 'values', parse the 'values', and save them.
;  519     0955  2  !
;  520     0956  2  If CLI$PRESENT (include_desc)
;  521     0957  2  Then
;  522     0958  2      !
;  523     0959  2      ! Indicate that the /include qualifier was specified, get
;  524     0960  2      ! any value(s) associated with it.
;  525     0961  2      !
;  526     0962  3      Begin
;  527     0963  3      Option_flag[opt$v_include_qual] = true ;
;  528     0964  3      Exclude_flag = false ;
;  529     0965  3
;  530     0966  3      While CLI$GET_VALUE (include_desc,wrk_desc) do
;  531     0967  4          Begin
;  532     0968  4          !
;  533     0969  4          ! Determine if the retrieved 'value' is a keyword
;  534     0970  4          ! or a device specification.
;  535     0971  4          !
;  536     0972  5          If NOT (GET_DEVICE_SELECT (wrk_desc))
;  537     0973  4          Then
;  538     0974  4              !
;  539     0975  4              ! Illegal input, notify the user and exit.
;  540     0976  4              !
;  541     0977  4              Signal_stop (msg$_invquaval, 2,
;  542     0978  4                           wrk_desc,include_desc) ;
;  543     0979  3          End ;
;  544     0980  2     End ;
;  545     0981  2
;  546     0982  2  !
;  547     0983  2  ! Determine whether the /LOG qualifier was specified.
;  548     0984  2  !
;  549     0985  2  If CLI$PRESENT (log_desc)
;  550     0986  2  Then
```

```
;  551     0987  2     !
;  552     0988  2     ! Indicate that it was speicified.
;  553     0989  2     !
;  554     0990  2     Option_flag[opt$v_log_qual] = true ;
;  555     0991  2
;  556     0992  2 !
;  557     0993  2 ! Determine whether the /PAGE qualifier was specified.
;  558     0994  2 !
;  559     0995  2 If CLI$PRESENT (page_desc)
;  560     0996  2 Then
;  561     0997  2     !
;  562     0998  2     ! Indicate that the qualifier was specified.
;  563     0999  2     !
;  564     1000  2     Option_flag[opt$v_page_qual] = true ;
;  565     1001  2
;  566     1002  2 !
;  567     1003  2 ! Determine whether the /REJECTED qualifier was specified. The
;  568     1004  2 ! file spec will be retrieved and parsed by the Parse_output_files
;  569     1005  2 ! routine in ERF.
;  570     1006  2 !
;  571     1007  2 If CLI$PRESENT (rejected_desc)
;  572     1008  2 Then
;  573     1009  2     !
;  574     1010  2     ! Indicate that the qualifier was specified.
;  575     1011  2     !
;  576     1012  2     Option_flag[opt$v_rejected_qual] = true ;
;  577     1013  2
;  578     1014  2 !
;  579     1015  2 ! Determine whether the /SID_REGISTER was specified.
;  580     1016  2 !
;  581     1017  2 If CLI$PRESENT (sid_register_desc)
;  582     1018  2 Then
;  583     1019  2     !
;  584     1020  2     ! Indicate that the qualifier was specified and get the
;  585     1021  2     ! value associated with it. The CLD will ensure that
;  586     1022  2     ! a value was specified.
;  587     1023  2     !
;  588     1024  3     Begin
;  589     1025  3     Option_flag[opt$v_sid_reg_qual] = true ;
;  590     1026  3     CLI$GET_VALUE (sid_register_desc,wrk_desc) ;
;  591     1027  3
;  592     1028  3     !
;  593     1029  3     ! Determine if the specified sid has characters outside the range
;  594     1030  3     ! of hexadecimal chars.
;  595     1031  3     !
;  596     1032  4     If (CH$GEQ (1,CH$PTR(.wrk_desc[dsc$a_pointer]),1,ptr_0) AND
;  597     1033  4         CH$LEQ (1,CH$PTR(.wrk_desc[dsc$a_pointer]),1,ptr_z))
;  598     1034  3     Then
;  599     1035  3         !
;  600     1036  3         ! Save the system id value.
;  601     1037  3         !
;  602     1038  3         Parser_data[erl$l_sid_selection] = .system_id ;
;  603     1039  2     End ;
;  604     1040  2
;  605     1041  2 !
;  606     1042  2 ! Indicate that the /SINCE qualifier was specified and get any value(s)
;  607     1043  2 ! associated with it.
```

```
;  608     1044  2 !
;  609     1045  2 If CLI$GET_VALUE (since_desc,wrk_desc)
;  610     1046  2 Then
;  611     1047  2     !
;  612     1048  2     ! Convert the ascii time/date string to binary time/date.
;  613     1049  2     ! The CLI will return a default value if the user did not specify one.
;  614     1050  2     ! LIB$CVT_TIME handles the Today, Yesterday, and Tomorrow keywords and
;  615     1051  2     ! will convert an absolute or delta time string or a combination of
;  616     1052  2     ! the two.
;  617     1053  2     !
;  618     1054  3     Begin
;  619     1055  3     Option_flag[opt$v_since_qual] = true ;
;  620     1056  4     If NOT (status = [IB$CVT_TIME (wrk_desc,parser_data[erl$q_start_date]))
;  621     1057  3     Then
;  622     1058  3         !
;  623     1059  3         ! Date/time conversion error, notify the user.
;  624     1060  3         !
;  625     1061  3         Signal (.status) ;
;  626     1062  2     End ;
;  627     1063  2
;  628     1064  2 !
;  629     1065  2 ! Determine whether the /STATISTICS qualifier was specified.
;  630     1066  2 !
;  631     1067  2 If CLI$PRESENT (statistics_desc)
;  632     1068  2 Then
;  633     1069  2     !
;  634     1070  2     ! Indicate that it was speicified.
;  635     1071  2     !
;  636     1072  2     Option_flag[opt$v_statistics_qual] = true ;
;  637     1073  2
;  638     1074  2 !
;  639     1075  2 ! Determine whether /SUMMARY was specified.
;  640     1076  2 !
;  641     1077  2 If CLI$PRESENT (summary_desc)
;  642     1078  2 Then
;  643     1079  2     !
;  644     1080  2     ! Indicate that a summary report was selected.
;  645     1081  2     !
;  646     1082  3     Begin
;  647     1083  3     Option_flag[opt$v_summary_qual] = true ;
;  648     1084  3
;  649     1085  3     !
;  650     1086  3     ! Get any value(s) associated with the qualifier.
;  651     1087  3     !
;  652     1088  3     While CLI$GET_VALUE (summary_desc,wrk_desc) do
;  653     1089  4         Begin
;  654     1090  4         !
;  655     1091  4         ! Get the value associated with the summary type keyword.
;  656     1092  4         !
;  657     1093  5         If (status = LIB$LOOKUP_KEY (wrk_desc,summary_keywords,key_value))
;  658     1094  4         Then
;  659     1095  4             !
;  660     1096  4             ! Indicate which summary options were selected.
;  661     1097  4             !
;  662     1098  5             Begin
;  663     1099  5
;  664     1100  5             Case .key_value from 1 to 5 of
```

```
;  665     1101  5                Set
;  666     1102  5                [1]:                        ! Device summary info
;  667     1103  6                    Begin
;  668     1104  6                    Summary_flag[sum$v_device] = true ;
;  669     1105  5                    End ;
;  670     1106  5
;  671     1107  5                [2]:                        ! Entry summary info
;  672     1108  6                    Begin
;  673     1109  6                    Summary_flag[sum$v_entry] = true ;
;  674     1110  5                    End ;
;  675     1111  5
;  676     1112  5                [3]:                        ! Memory summary info
;  677     1113  6                    Begin
;  678     1114  6                    Summary_flag[sum$v_memory] = true ;
;  679     1115  5                    End ;
;  680     1116  5
;  681     1117  5                [4]:                        ! Volume summary info
;  682     1118  6                    Begin
;  683     1119  6                    Summary_flag[sum$v_volume] = true ;
;  684     1120  5                    End ;
;  685     1121  5
;  686     1122  5                [5]:                        ! Histogram summary info
;  687     1123  6                    Begin
;  688     1124  6                    Summary_flag[sum$v_histogram] = true ;
;  689     1125  5                    End ;
;  690     1126  5
;  691     1127  5                Tes ;
;  692     1128  5            End
;  693     1129  4        Else
;  694     1130  4            !
;  695     1131  4            ! Illegal input, notify the user.
;  696     1132  4            !
;  697     1133  4            Signal_stop (msg$_invquaval, 2,wrk_desc,summary_desc) ;
;  698     1134  3            End ;
;  699     1135  3
;  700     1136  3        If ..summary_flag EQL 0
;  701     1137  3        Then summary_flag[sum$v_all_summ] = true ;
;  702     1138  3
;  703     1139  2        End ;
;  704     1140  2
;  705     1141  2 !
;  706     1142  2 ! Determine if the /INCLUDE or /EXCLUDE qualifiers were specified and
;  707     1143  2 ! set up the defaults for what to output.
;  708     1144  2 !
;  709     1145  2 If (NOT .option_flag[opt$v_include_qual]) AND
;  710     1146  3     (NOT .option_flag[opt$v_exclude_qual])
;  711     1147  2 Then
;  712     1148  2     !
;  713     1149  2     ! Default to outputting of all entry types,
;  714     1150  2     ! device classes, and device names.
;  715     1151  2     !
;  716     1152  3     Begin
;  717     1153  3     Option_flag[opt$v_output_all] = true ;
;  718     1154  3     Include_mask[inc$v_device_select] = false ;
;  719     1155  3     Exclude_mask[exc$v_device_select] = false ;
;  720     1156  3     Include_mask[inc$v_dev_class_select] = false ;
;  721     1157  3     Exclude_mask[exc$v_dev_class_select] = false ;
```

```
;  722     1158  3     Include_mask[inc$v_entry_select] = false ;
;  723     1159  3     Exclude_mask[exc$v_entry_select] = false ;
;  724     1160  3     Include_mask[inc$v_unknown_entry] = false ;
;  725     1161  3     Exclude_mask[exc$v_unknown_entry] = false ;
;  726     1162  2     End ;
;  727     1163  2
;  728     1164  2 !
;  729     1165  2 ! Determine if the /FULL report type qualifier was
;  730     1166  2 ! specified.
;  731     1167  2 !
;  732     1168  2 Status = CLI$PRESENT (full_desc) ;
;  733     1169  2 If .status EQL cli$_present
;  734     1170  2 Then
;  735     1171  2     !
;  736     1172  2     ! Indicate the report type.
;  737     1173  2     !
;  738     1174  3     Begin
;  739     1175  3     I  = .I + 1 ;
;  740     1176  3     Parser_data[erl$b_rpt_type] = full_rep ;
;  741     1177  3     End
;  742     1178  2 Else
;  743     1179  3     Begin
;  744     1180  3     If .status EQL cli$_negated
;  745     1181  3     Then
;  746     1182  3         !
;  747     1183  3         ! Indicate that /NOFULL was specified.
;  748     1184  3         !
;  749     1185  4         Begin
;  750     1186  4         Parser_data[erl$b_rpt_type] = 0 ;
;  751     1187  4         Full_negate = true ;
;  752     1188  3         End ;
;  753     1189  2     End ;
;  754     1190  2
;  755     1191  2 !
;  756     1192  2 ! Determine if the /BRIEF report type qualifier was
;  757     1193  2 ! specified.
;  758     1194  2 !
;  759     1195  2 If CLI$PRESENT (brief_desc)
;  760     1196  2 Then
;  761     1197  2     !
;  762     1198  2     ! Indicate the report type.
;  763     1199  2     !
;  764     1200  3     Begin
;  765     1201  3     I  = .I + 1 ;
;  766     1202  3     Parser_data[erl$b_rpt_type] = brief_rep ;
;  767     1203  2     End ;
;  768     1204  2
;  769     1205  2 !
;  770     1206  2 ! Determine if the /REGISTER_DUMP report type qualifier was
;  771     1207  2 ! specified.
;  772     1208  2 !
;  773     1209  2 If CLI$PRESENT (register_dump_desc)
;  774     1210  2 Then
;  775     1211  2     !
;  776     1212  2     ! Indicate that the report type and ensure that device
;  777     1213  2     ! selection was made.
;  778     1214  2     !
```

```
;  779     1215  3     Begin
;  780     1216  3     I  = .I + 1 ;
;  781     1217  3     Parser_data[erl$b_rpt_type] = reg_dump_rep ;
;  782     1218  3
;  783     1219  3     If (NOT .option_flag[opt$v_include_qual]) OR
;  784     1220  4        (dev_entry_key)
;  785     1221  3     Then
;  786     1222  3          !
;  787     1223  3          ! Either a device was not selected or an invalid
;  788     1224  3          ! device selection was made, notify the user and exit.
;  789     1225  3          !
;  790     1226  3          Signal_stop (erf_devselreq) ;
;  791     1227  2     End ;
;  792     1228  2
;  793     1229  2
;  794     1230  2  !
;  795     1231  2  ! Determine if a report type was specified and ensure that the
;  796     1232  2  ! /BINARY qualifier was not specified also.
;  797     1233  2  !
;  798     1234  3  If (.parser_data[erl$b_rpt_type] NEQ 0)
;  799     1235  2  Then
;  800     1236  2       !
;  801     1237  2       ! Ensure that the /binary qualifier was not specified.
;  802     1238  2       !
;  803     1239  3       Begin
;  804     1240  3       If .option_flag[opt$v_binary_qual]
;  805     1241  3       Then
;  806     1242  3            !
;  807     1243  3            ! Illegal combination of qualifiers, notify the user and
;  808     1244  3            ! exit.
;  809     1245  3            !
;  810     1246  3            Signal_stop (msg$_confqual) ;
;  811     1247  2       End ;
;  812     1248  2
;  813     1249  2  !
;  814     1250  2  ! Ensure that there was only one report type specified.
;  815     1251  2  !
;  816     1252  2  If .I GTR 1
;  817     1253  2  Then
;  818     1254  2       !
;  819     1255  2       ! Illegal combination of qualifiers, notify the user and exit.
;  820     1256  2       !
;  821     1257  2       Signal_stop (msg$_confqual) ;
;  822     1258  2
;  823     1259  2  !
;  824     1260  2  ! Determine if there are any conflicts between /exclude
;  825     1261  2  ! and /include device class option selections.
;  826     1262  2  ! Do not look at the device, device class, or entry selection
;  827     1263  2  ! indicators.
;  828     1264  2  !
;  829     1265  2  Status = (..include_mask AND .keywrd_mask) AND
;  830     1266  2           (..exclude_mask AND .keywrd_mask);
;  831     1267  2
;  832     1268  2  If .status NEQU 0
;  833     1269  2  Then
;  834     1270  2       !
;  835     1271  2       ! Illegal combination of /exclude and /include
```

```
;  836          1272  2     ! options, notify the user and exit.
;  837          1273  2     !
;  838          1274  2     Signal_stop (erf_cnfquaval, 2,exclude_desc,include_desc) ;
;  839          1275  2
;  840          1276  2 !
;  841          1277  2 ! Determine if there are any conflicts between any of the
;  842          1278  2 ! selected devices and any selected device class options
;  843          1279  2 ! for /include and /exclude.
;  844          1280  2 !
;  845          1281  2 CLASS_OPTION_CHECK () ;
;  846          1282  2
;  847          1283  2 !
;  848          1284  2 ! Indicate that the command line is parsed, by returning
;  849          1285  2 ! to the calling routine with a true value.
;  850          1286  2 !
;  851          1287  2 Return true ;
;  852          1288  1 End ;              ! Routine


                                                        .TITLE  ERFPARSER Command Parser
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT,NOWRT,NOEXE,  PIC,2

                                45  4E  49  4C  24  00000 P.AAB:  .ASCII  \$LINE\                          ;
                                                    00005       .BLKB   3
                                      00000005      00008 P.AAA:  .LONG   5                                 ;
                                      00000000'     0000C         .ADDRESS P.AAB                            ;
                            53  43  45  50  53  5F  45  4C  49  46  00010 P.AAD:  .ASCII  \FILE_SPECS\      ;
                                                    0001A         .BLKB   2
                                      0000000A      0001C P.AAC:  .LONG   10                                ;
                                      00000000'     00020         .ADDRESS P.AAD                            ;
                                    45  52  4F  46  45  42  00024 P.AAF:  .ASCII  \BEFORE\                 ;
                                                    0002A         .BLKB   2
                                      00000006      0002C P.AAE:  .LONG   6                                 ;
                                      00000000'     00030         .ADDRESS P.AAF                            ;
                                    59  52  41  4E  49  42  00034 P.AAH:  .ASCII  \BINARY\                 ;
                                                    0003A         .BLKB   2
                                      00000006      0003C P.AAG:  .LONG   6                                 ;
                                      00000000'     00040         .ADDRESS P.AAH                            ;
                                        46  45  49  52  42  00044 P.AAJ:  .ASCII  \BRIEF\                   ;
                                                    00049         .BLKB   3
                                      00000005      0004C P.AAI:  .LONG   5                                 ;
                                      00000000'     00050         .ADDRESS P.AAJ                            ;
                                        59  52  54  4E  45  00054 P.AAL:  .ASCII  \ENTRY\                   ;
                                                    00059         .BLKB   3
                                      00000005      0005C P.AAK:  .LONG   5                                 ;
                                      00000000'     00060         .ADDRESS P.AAL                            ;
                                45  44  55  4C  43  58  45  00064 P.AAN:  .ASCII  \EXCLUDE\                 ;
                                                    0006B         .BLKB   1
                                      00000007      0006C P.AAM:  .LONG   7                                 ;
                                      00000000'     00070         .ADDRESS P.AAN                            ;
                                            4C  4C  55  46  00074 P.AAP:  .ASCII  \FULL\                   ;
                                      00000004      00078 P.AAO:  .LONG   4                                 ;
                                      00000000'     0007C         .ADDRESS P.AAP                            ;
                                45  44  55  4C  43  4E  49  00080 P.AAR:  .ASCII  \INCLUDE\                 ;
                                                    00087         .BLKB   1
```

```
                                    00000007  00088 P.AAQ:  .LONG    7                          ;
                                    00000000' 0008C         .ADDRESS P.AAR                      ;
                                  47 4F 4C    00090 P.AAT:  .ASCII   \LOG\                      ;
                                              00093         .BLKB    1
                                    00000003  00094 P.AAS:  .LONG    3                          ;
                                    00000000' 00098         .ADDRESS P.AAT                      ;
                        54 55 50 54 55 4F     0009C P.AAV:  .ASCII   \OUTPUT\                   ;
                                              000A2         .BLKB    2
                                    00000006  000A4 P.AAU:  .LONG    6                          ;
                                    00000000' 000A8         .ADDRESS P.AAV                      ;
                               45 47 41 50    000AC P.AAX:  .ASCII   \PAGE\                     ;
                                    00000004  000B0 P.AAW:  .LONG    4                          ;
                                    00000000' 000B4         .ADDRESS P.AAX                      ;
   50 4D 55 44 5F 52 45 54 53 49 47 45 52     000B8 P.AAZ:  .ASCII   \REGISTER_DUMP\            ;
                                              000C5         .BLKB    3
                                    0000000D  000C8 P.AAY:  .LONG    13                         ;
                                    00000000' 000CC         .ADDRESS P.AAZ                      ;
                  44 45 54 43 45 4A 45 52     000D0 P.ABB:  .ASCII   \REJECTED\                 ;
                                    00000008  000D8 P.ABA:  .LONG    8                          ;
                                    00000000' 000DC         .ADDRESS P.ABB                      ;
      52 45 54 53 49 47 45 52 5F 44 49 53     000E0 P.ABD:  .ASCII   \SID_REGISTER\             ;
                                    0000000C  000EC P.ABC:  .LONG    12                         ;
                                    00000000' 000F0         .ADDRESS P.ABD                      ;
                           45 43 4E 49 53     000F4 P.ABF:  .ASCII   \SINCE\                    ;
                                              000F9         .BLKB    3
                                    00000005  000FC P.ABE:  .LONG    5                          ;
                                    00000000' 00100         .ADDRESS P.ABF                      ;
            53 43 49 54 53 49 54 41 54 53     00104 P.ABH:  .ASCII   \STATISTICS\               ;
                                              0010E         .BLKB    2
                                    0000000A  00110 P.ABG:  .LONG    10                         ;
                                    00000000' 00114         .ADDRESS P.ABH                      ;
                     59 52 41 4D 4D 55 53     00118 P.ABJ:  .ASCII   \SUMMARY\                  ;
                                              0011F         .BLKB    1
                                    00000007  00120 P.ABI:  .LONG    7                          ;
                                    00000000' 00124         .ADDRESS P.ABJ                      ;
               44 4E 45 2E 59 52 54 4E 45     00128 P.ABL:  .ASCII   \ENTRY.END\                ;
                                              00131         .BLKB    3
                                    00000009  00134 P.ABK:  .LONG    9                          ;
                                    00000000' 00138         .ADDRESS P.ABL                      ;
         54 52 41 54 53 2E 59 52 54 4E 45     0013C P.ABN:  .ASCII   \ENTRY.START\              ;
                                              00147         .BLKB    1
                                    0000000B  00148 P.ABM:  .LONG    11                         ;
                                    00000000' 0014C         .ADDRESS P.ABN                      ;
                               00 00 00 30    00150 P.ABO:  .ASCII   \0\<0><0><0>               ;
                               00 00 00 7A    00154 P.ABP:  .ASCII   \z\<0><0><0>               ;
                   00 45 43 49 56 45 44 06    00158 P.ABQ:  .ASCII   <6>\DEVICE\<0>             ;
                   00 00 59 52 54 4E 45 05    00160 P.ABR:  .ASCII   <5>\ENTRY\<0><0>           ;
                   00 59 52 4F 4D 45 4D 06    00168 P.ABS:  .ASCII   <6>\MEMORY\<0>             ;
                   00 45 4D 55 4C 4F 56 06    00170 P.ABT:  .ASCII   <6>\VOLUME\<0>             ;
         00 00 4D 41 52 47 4F 54 53 49 48 09  00178 P.ABU:  .ASCII   <9>\HISTOGRAM\<0><0>       ;

                                                            .PSECT  QUEUE_DATA,NOEXE,  PIC,3

                                    00000000  00000 ROOT_FLINK::
                                                            .LONG    0                          ;
                                    00000000  00004 ROOT_BLINK::
                                                            .LONG    0                          ;
```

```
                        00000000  00008 QUE_ADDRS::
                                          .LONG   0                                       ;
                        00000000  0000C QUE_ENTRY_ADDRS::
                                          .LONG   0                                       ;

                                          .PSECT  $OWN$,NOEXE,  PIC,2

                        000FFFFF  00000 KEYWRD_MASK:
                                          .LONG   1048575                                 ;
                              00# 00004 WRK_DESC:
                                          .BYTE   0[3]                                    ;
                              02  00007   .BYTE   2                                       ;
                                  00008   .BLKB   4
                        0000000A  0000C SUMMARY_KEYWORDS:
                                          .LONG   10                                      ;
                        00000000' 00010   .ADDRESS P.ABQ                                  ;
                        00000001  00014   .LONG   1                                       ;
                        00000000' 00018   .ADDRESS P.ABR                                  ;
                        00000002  0001C   .LONG   2                                       ;
                        00000000' 00020   .ADDRESS P.ABS                                  ;
                        00000003  00024   .LONG   3                                       ;
                        00000000' 00028   .ADDRESS P.ABT                                  ;
                        00000004  0002C   .LONG   4                                       ;
                        00000000' 00030   .ADDRESS P.ABU                                  ;
                        00000005  00034   .LONG   5                                       ;

                                          .PSECT  $GLOBAL$,NOEXE,  PIC,2

                                  00000 DEV_CLASS_KEY::
                                          .BLKB   4
                                  00004 DEV_ENTRY_KEY::
                                          .BLKB   4
                                  00008 DEV_NAME::
                                          .BLKB   4
                                  0000C DEV_SELECT::
                                          .BLKB   4
                                  00010 ENTRY_VALUE::
                                          .BLKB   4
                                  00014 EXCLUDE_CLASS::
                                          .BLKB   6
                                  0001A EXCLUDE_FLAG::
                                          .BLKB   1
                                  0001B   .BLKB   1
                                  0001C EXCLUDE_KEY::
                                          .BLKB   6
                                  00022   .BLKB   2
                                  00024 EXCLUDE_MASK::
                                          .BLKB   4
                                  00028 EXCLUDE_Q_ENTRY_CNT::
                                          .BLKB   1
                                  00029 INCLUDE_Q_ENTRY_CNT::
                                          .BLKB   1
                                  0002A   .BLKB   2
                                  0002C INCLUDE_CLASS::
                                          .BLKB   6
                                  00032   .BLKB   2
                                  00034 INCLUDE_KEY::
```

```
                        .BLKB   6
0003A                   .BLKB   2
0003C INCLUDE_MASK::
                        .BLKB   4
00040 OPTION_FLAG::
                        .BLKB   4
00044 PARSER_DATA::
                        .BLKB   4
00048 PARSER_TABLE::
                        .BLKB   4
0004C QUE_ENTRY_CNT::
                        .BLKB   2
0004E                   .BLKB   2
00050 SUMMARY_FLAG::
                        .BLKB   4
00054 CLASS_DIR::
                        .BLKB   4
00058 WILD_CARDED_DEVICE::
                        .BLKB   4

      BUS==                     1
      DISK==                    2
      REALTIME==                3
      SYNC==                    4
      TAPE==                    5
      ERROR==                   2
      MAX_CLASS==               5
      Q_ENTRY_SIZE==            32
      $CLINE_DESC==             P.AAA
      FILE_SPECS_DESC==         P.AAC
      BEFORE_DESC==             P.AAE
      BINARY_DESC==             P.AAG
      BRIEF_DESC==              P.AAI
      ENTRY_DESC==              P.AAK
      EXCLUDE_DESC==            P.AAM
      FULL_DESC==               P.AAO
      INCLUDE_DESC==            P.AAQ
      LOG_DESC==                P.AAS
      OUTPUT_DESC==             P.AAU
      PAGE_DESC==               P.AAW
      REGISTER_DUMP_DESC==      P.AAY
      REJECTED_DESC==           P.ABA
      SID_REGISTER_DESC==       P.ABC
      SINCE_DESC==              P.ABE
      STATISTICS_DESC==         P.ABG
      SUMMARY_DESC==            P.ABI
      ENTRY_END_DESC=           P.ABK
      ENTRY_START_DESC=         P.ABM
      PTR_O=                    P.ABO
      PTR_Z=                    P.ABP
              .EXTRN  CLI$GET_VALUE, CLI$PRESENT
              .EXTRN  LIB$LOOKUP_KEY, LIB$CVT_TIME
              .EXTRN  LIB$CVT_DTB, LIB$CVT_HTB
              .EXTRN  LIB$GET_VM, LIB$INSQTI
              .EXTRN  LIB$REMQTI, CLI$_ABSENT
              .EXTRN  CLI$_NEGATED, CLI$_PRESENT
              .EXTRN  ERF_CNFQUAVAL, ERF_CVTERR
```

```
                                        .EXTRN  ERF_DEVSELREQ

                                        .PSECT  $CODE,NOWRT,  PIC,2

                          0FFC 00000    .ENTRY  PARSE_COMMAND, Save R2,R3,R4,R5,R6,R7,R8,-      ; 0693
                                                R9,R10,R11
                5B 00000000V   00  9E 00002    MOVAB   GET_VM, R11
                5A 00000000G   00  9E 00009    MOVAB   LIB$STOP, R10
                59 00000000G   00  9E 00010    MOVAB   CLI$GET_VALUE, R9
                58 00000000G   00  9E 00017    MOVAB   CLI$PRESENT, R8
                57 00000000'   00  9E 0001E    MOVAB   WRK_DESC, R7
                56 00000000'   00  9E 00025    MOVAB   EXCLUDE_DESC, R6
                55 00000000'   00  9E 0002C    MOVAB   OPTION_FLAG, R5
                5E                 04  C2 00033    SUBL2   #4, SP
                                   57  7C 00036    CLRQ    I                                   ; 0715
                                   08  DD 00038    PUSHL   #8                                  ; 0772
                6B                 01  FB 0003A    CALLS   #1, GET_VM
          08    A5                 50  D0 0003D    MOVL    R0, PARSER_TABLE
                7E          51     8F  9A 00041    MOVZBL  #81, -(SP)                          ; 0773
                6B                 01  FB 00045    CALLS   #1, GET_VM
          04    A5                 50  D0 00048    MOVL    R0, PARSER_DATA
                60                 02  90 0004C    MOVB    #2, (R0)                            ; 0778
          05    A0                 01  CE 0004F    MNEGL   #1, 5(R0)                           ; 0779
          09    A0 7FFFFFFF        8F  D0 00053    MOVL    #2147483647, 9(R0)                  ; 0780
                          0D       A0  7C 0005B    CLRQ    13(R0)                              ; 0781
                          15       A0  D4 0005E    CLRL    21(R0)                              ; 0784
          19    A0                 01  CE 00061    MNEGL   #1, 25(R0)                          ; 0785
                                   03  DD 00065    PUSHL   #3                                  ; 0794
                6B                 01  FB 00067    CALLS   #1, GET_VM
          E4    A5                 50  D0 0006A    MOVL    R0, EXCLUDE_MASK
                                   03  DD 0006E    PUSHL   #3                                  ; 0798
                6B                 01  FB 00070    CALLS   #1, GET_VM
                FC    A5           50  D0 00073    MOVL    R0, INCLUDE_MASK
                                   03  DD 00077    PUSHL   #3                                  ; 0803
                6B                 01  FB 00079    CALLS   #1, GET_VM
                65                 50  D0 0007C    MOVL    R0, OPTION_FLAG
                60          0120   8F  A8 0007F    BISW2   #288, (R0)                          ; 0804
                                   01  DD 00084    PUSHL   #1                                  ; 0809
                6B                 01  FB 00086    CALLS   #1, GET_VM
                10    A5           50  D0 00089    MOVL    R0, SUMMARY_FLAG
                50          08     A5  D0 0008D    MOVL    PARSER_TABLE, R0                    ; 0813
                03    A0           02  90 00091    MOVB    #2, 3(R0)
                                   50  DD 00095    PUSHL   CMD_LINE_DESC                       ; 0820
                          9C       A6  9F 00097    PUSHAB  $LINE_DESC
                69                 02  FB 0009A    CALLS   #2, CLI$GET_VALUE
                          C0       A6  9F 0009D    PUSHAB  BEFORE_DESC                         ; 0827
                68                 01  FB 000A0    CALLS   #1, CLI$PRESENT
                2E                 50  E9 000A3    BLBC    R0, 1$
                50                 65  D0 000A6    MOVL    OPTION_FLAG, R0                     ; 0833
                60                 01  88 000A9    BISB2   #1, (R0)
                                   57  DD 000AC    PUSHL   R7                                  ; 0834
                          C0       A6  9F 000AE    PUSHAB  BEFORE_DESC
                69                 02  FB 000B1    CALLS   #2, CLI$GET_VALUE
                1D                 50  E9 000B4    BLBC    R0, 1$
          7E          04    A5     05  C1 000B7    ADDL3   #5, PARSER_DATA, -(SP)              ; 0843
                                   57  DD 000BC    PUSHL   R7
     00000000G    00                 02  FB 000BE    CALLS   #2, LIB$CVT_TIME
```

```
                        52              50  D0 000C5          MOVL    R0, STATUS                              ;
                        09              52  E8 000C8          BLBS    STATUS, 1$                              ;
                                        52  DD 000CB          PUSHL   STATUS                                  ; 0848
              00000000G 00              01  FB 000CD          CALLS   #1, LIB$SIGNAL                          ;
                                  D0    A6  9F 000D4 1$:      PUSHAB  BINARY_DESC                             ; 0857
                        68              01  FB 000D7          CALLS   #1, CLI$PRESENT                         ;
                        2A              50  E9 000DA          BLBC    R0, 3$                                  ;
                                  38    A6  9F 000DD          PUSHAB  OUTPUT_DESC                             ; 0864
                        68              01  FB 000E0          CALLS   #1, CLI$PRESENT                         ;
                        52              50  D0 000E3          MOVL    R0, STATUS                              ;
              00000000G 8F              52  D1 000E6          CMPL    STATUS, #CLI$_PRESENT                   ; 0865
                                        09  12 000ED          BNEQ    2$                                      ;
                              000812E3  8F  DD 000EF          PUSHL   #529123                                 ; 0867
                        6A              01  FB 000F5          CALLS   #1, LIB$STOP                            ;
                        50              65  D0 000F8 2$:      MOVL    OPTION_FLAG, R0                         ; 0869
                        60              02  88 000FB          BISB2   #2, (R0)                                ;
                        60              20  8A 000FE          BICB2   #32, (R0)                               ; 0870
                        50        04    A5  D0 00101          MOVL    PARSER_DATA, R0                         ; 0871
                                        60  94 00105          CLRB    (R0)                                    ;
                                  F0    A6  9F 00107 3$:      PUSHAB  ENTRY_DESC                              ; 0878
                        68              01  FB 0010A          CALLS   #1, CLI$PRESENT                         ;
                        4F              50  E9 0010D          BLBC    R0, 6$                                  ;
                        50              65  D0 00110          MOVL    OPTION_FLAG, R0                         ; 0885
                        60              08  88 00113          BISB2   #8, (R0)                                ;
                                        57  DD 00116          PUSHL   R7                                      ; 0891
                                  00DC  C6  9F 00118          PUSHAB  ENTRY_START_DESC                        ;
                        69              02  FB 0011C          CALLS   #2, CLI$GET_VALUE                       ;
                        34              50  E9 0011F          BLBC    R0, 5$                                  ;
                                  D0    A5  9F 00122          PUSHAB  ENTRY_VALUE                             ; 0898
                                  04    A7  DD 00125          PUSHL   WRK_DESC+4                              ;
                              7E        67  3C 00128          MOVZWL  WRK_DESC, -(SP)                         ;
              00000000G       00        03  FB 0012B          CALLS   #3, LIB$CVT_DTB                         ;
                        52              50  D0 00132          MOVL    R0, STATUS                              ;
                        15              52  E8 00135          BLBS    STATUS, 4$                              ;
                                  04    A7  DD 00138          PUSHL   WRK_DESC+4                              ;
                              7E        67  3C 0013B          MOVZWL  WRK_DESC, -(SP)                         ;
                                        02  DD 0013E          PUSHL   #2                                      ;
                          00000000G     8F  DD 00140          PUSHL   #ERF_CVTERR                             ;
              00000000G       00        04  FB 00146          CALLS   #4, LIB$SIGNAL                          ;
                        50        04    A5  D0 0014D 4$:      MOVL    PARSER_DATA, R0                         ; 0900
                     15 A0        D0    A5  D0 00151          MOVL    ENTRY_VALUE, 21(R0)                     ;
                                        57  DD 00156 5$:      PUSHL   R7                                      ; 0906
                                  00C8  C6  9F 00158          PUSHAB  ENTRY_END_DESC                          ;
                        69              02  FB 0015C          CALLS   #2, CLI$GET_VALUE                       ;
                        34              50  E9 0015F 6$:      BLBC    R0, 8$                                  ;
                                  D0    A5  9F 00162          PUSHAB  ENTRY_VALUE                             ; 0913
                                  04    A7  DD 00165          PUSHL   WRK_DESC+4                              ;
                              7E        67  3C 00168          MOVZWL  WRK_DESC, -(SP)                         ;
              00000000G       00        03  FB 0016B          CALLS   #3, LIB$CVT_DTB                         ;
                        52              50  D0 00172          MOVL    R0, STATUS                              ;
                        15              52  E8 00175          BLBS    STATUS, 7$                              ;
                                  04    A7  DD 00178          PUSHL   WRK_DESC+4                              ;
                              7E        67  3C 0017B          MOVZWL  WRK_DESC, -(SP)                         ;
                                        02  DD 0017E          PUSHL   #2                                      ;
                          00000000G     8F  DD 00180          PUSHL   #ERF_CVTERR                             ;
              00000000G       00        04  FB 00186          CALLS   #4, LIB$SIGNAL                          ;
                        50        04    A5  D0 0018D 7$:      MOVL    PARSER_DATA, R0                         ; 0915
```

```
        19  A0        D0  A5  D0 00191        MOVL    ENTRY_VALUE, 25(R0)
                          56  DD 00196 8$:    PUSHL   R6                          ; 0922
            68            01  FB 00198        CALLS   #1, CLI$PRESENT             ;
            31            50  E9 0019B        BLBC    R0, 10$                     ;
            50            65  D0 0019E        MOVL    OPTION_FLAG, R0             ; 0928
            60            10  88 001A1        BISB2   #16, (R0)                   ;
        DA  A5            01  90 001A4        MOVB    #1, EXCLUDE_FLAG            ; 0929
                  00C0    8F  BB 001A8 9$:    PUSHR   #^M<R6,R7>                  ; 0934
            69            02  FB 001AC        CALLS   #2, CLI$GET_VALUE           ;
            1D            50  E9 001AF        BLBC    R0, 10$                     ;
                          57  DD 001B2        PUSHL   R7                          ; 0939
 00000000V  00            01  FB 001B4        CALLS   #1, GET_DEVICE_SELECT       ;
            EA            50  E8 001BB        BLBS    R0, 9$                      ;
                          56  DD 001BE        PUSHL   R6                          ; 0945
                          57  DD 001C0        PUSHL   R7                          ;
                          02  DD 001C2        PUSHL   #2                          ;
              0008132C    8F  DD 001C4        PUSHL   #529196                     ;
            6A            04  FB 001CA        CALLS   #4, LIB$STOP                ;
                          D9  11 001CD        BRB     9$                          ; 0934
                    1C    A6  9F 001CF 10$:   PUSHAB  INCLUDE_DESC                ; 0956
            68            01  FB 001D2        CALLS   #1, CLI$PRESENT             ;
            33            50  E9 001D5        BLBC    R0, 12$                     ;
            50            65  D0 001D8        MOVL    OPTION_FLAG, R0             ; 0963
            60      40    8F  88 001DB        BISB2   #64, (R0)                   ;
                    DA    A5  94 001DF        CLRB    EXCLUDE_FLAG                ; 0964
                          57  DD 001E2 11$:   PUSHL   R7                          ; 0966
                    1C    A6  9F 001E4        PUSHAB  INCLUDE_DESC                ;
            69            02  FB 001E7        CALLS   #2, CLI$GET_VALUE           ;
            1E            50  E9 001EA        BLBC    R0, 12$                     ;
                          57  DD 001ED        PUSHL   R7                          ; 0972
 00000000V  00            01  FB 001EF        CALLS   #1, GET_DEVICE_SELECT       ;
            E9            50  E8 001F6        BLBS    R0, 11$                     ;
                    1C    A6  9F 001F9        PUSHAB  INCLUDE_DESC                ; 0977
                          57  DD 001FC        PUSHL   R7                          ;
                          02  DD 001FE        PUSHL   #2                          ;
              0008132C    8F  DD 00200        PUSHL   #529196                     ;
            6A            04  FB 00206        CALLS   #4, LIB$STOP                ;
                          D7  11 00209        BRB     11$                         ; 0966
                    28    A6  9F 0020B 12$:   PUSHAB  LOG_DESC                    ; 0985
            68            01  FB 0020E        CALLS   #1, CLI$PRESENT             ;
            07            50  E9 00211        BLBC    R0, 13$                     ;
            50            65  D0 00214        MOVL    OPTION_FLAG, R0             ; 0990
            60      80    8F  88 00217        BISB2   #128, (R0)                  ;
                    44    A6  9F 0021B 13$:   PUSHAB  PAGE_DESC                   ; 0995
            68            01  FB 0021E        CALLS   #1, CLI$PRESENT             ;
            07            50  E9 00221        BLBC    R0, 14$                     ;
            50            65  D0 00224        MOVL    OPTION_FLAG, R0             ; 1000
        01  A0            02  88 00227        BISB2   #2, 1(R0)                   ;
                    6C    A6  9F 0022B 14$:   PUSHAB  REJECTED_DESC               ; 1007
            68            01  FB 0022E        CALLS   #1, CLI$PRESENT             ;
            07            50  E9 00231        BLBC    R0, 15$                     ;
            50            65  D0 00234        MOVL    OPTION_FLAG, R0             ; 1012
        01  A0            04  88 00237        BISB2   #4, 1(R0)                   ;
                  0080    C6  9F 0023B 15$:   PUSHAB  SID_REGISTER_DESC           ; 1017
            68            01  FB 0023F        CALLS   #1, CLI$PRESENT             ;
            2A            50  E9 00242        BLBC    R0, 16$                     ;
            50            65  D0 00245        MOVL    OPTION_FLAG, R0             ; 1025
```

```
                         01   A0        10  88 00248          BISB2   #16, 1(R0)                        ;
                                        57  DD 0024C          PUSHL   R7                                ; 1026
                                  0080  C6  9F 0024E          PUSHAB  SID_REGISTER_DESC                 ;
                              69        02  FB 00252          CALLS   #2, CLI$GET_VALUE                 ;
                              50    04  A7  D0 00255          MOVL    WRK_DESC+4, R0                    ; 1032
                       00E4   C6        60  91 00259          CMPB    (R0), PTR_0                       ;
                                        0F  1F 0025E          BLSSU   16$                               ;
                       00E8   C6        60  91 00260          CMPB    (R0), PTR_Z                       ; 1033
                                        08  1A 00265          BGTRU   16$                               ;
                              50    04  A5  D0 00267          MOVL    PARSER_DATA, R0                   ; 1038
                         01   A0        54  D0 0026B          MOVL    SYSTEM_ID, 1(R0)                  ;
                                        57  DD 0026F  16$:    PUSHL   R7                                ; 1045
                                  0090  C6  9F 00271          PUSHAB  SINCE_DESC                        ;
                              69        02  FB 00275          CALLS   #2, CLI$GET_VALUE                 ;
                              24        50  E9 00278          BLBC    R0, 17$                           ;
                              50        65  D0 0027B          MOVL    OPTION_FLAG, R0                   ; 1055
                         01   A0        20  88 0027E          BISB2   #32, 1(R0)                        ;
                 7E      04   A5        0D  C1 00282          ADDL3   #13, PARSER_DATA, -(SP)           ; 1056
                                        57  DD 00287          PUSHL   R7                                ;
                  00000000G   00        02  FB 00289          CALLS   #2, LIB$CVT_TIME                  ;
                              52        50  D0 00290          MOVL    R0, STATUS                        ;
                              09        52  E8 00293          BLBS    STATUS, 17$                       ;
                                        52  DD 00296          PUSHL   STATUS                            ; 1061
                  00000000G   00        01  FB 00298          CALLS   #1, LIB$SIGNAL                    ;
                                  00A4  C6  9F 0029F  17$:    PUSHAB  STATISTICS_DESC                   ; 1067
                              68        01  FB 002A3          CALLS   #1, CLI$PRESENT                   ;
                              07        50  E9 002A6          BLBC    R0, 18$                           ;
                              50        65  D0 002A9          MOVL    OPTION_FLAG, R0                   ; 1072
                         02   A0        01  88 002AC          BISB2   #1, 2(R0)                         ;
                                  00B4  C6  9F 002B0  18$:    PUSHAB  SUMMARY_DESC                      ; 1077
                              68        01  FB 002B4          CALLS   #1, CLI$PRESENT                   ;
                              71        50  E9 002B7          BLBC    R0, 28$                           ;
                              50        65  D0 002BA          MOVL    OPTION_FLAG, R0                   ; 1083
                         01   A0    40  8F  88 002BD          BISB2   #64, 1(R0)                        ;
                                        57  DD 002C2  19$:    PUSHL   R7                                ; 1088
                                  00B4  C6  9F 002C4          PUSHAB  SUMMARY_DESC                      ;
                              69        02  FB 002C8          CALLS   #2, CLI$GET_VALUE                 ;
                              52        50  E9 002CB          BLBC    R0, 27$                           ;
                                        5E  DD 002CE          PUSHL   SP                                ; 1093
                                    08  A7  9F 002D0          PUSHAB  SUMMARY_KEYWORDS                  ;
                                        57  DD 002D3          PUSHL   R7                                ;
                  00000000G   00        03  FB 002D5          CALLS   #3, LIB$LOOKUP_KEY                ;
                              52        50  D0 002DC          MOVL    R0, STATUS                        ;
                              2B        52  E9 002DF          BLBC    STATUS, 26$                       ;
                              50    10  A5  D0 002E2          MOVL    SUMMARY_FLAG, R0                  ; 1104
                 04           01        6E  CF 002E6          CASEL   KEY_VALUE, #1, #4                 ; 1100
      0019     0014         000F      000A     002EA  20$:    .WORD   21$-20$,-                         ;
                                      001E     002F2                  22$-20$,-                         ;
                                                                      23$-20$,-                         ;
                                                                      24$-20$,-                         ;
                                                                      25$-20$                           ;
                              60        02  88 002F4  21$:    BISB2   #2, (R0)                          ; 1104
                                        C9  11 002F7          BRB     19$                               ; 1100
                              60        04  88 002F9  22$:    BISB2   #4, (R0)                          ; 1109
                                        C4  11 002FC          BRB     19$                               ; 1100
                              60        08  88 002FE  23$:    BISB2   #8, (R0)                          ; 1114
                                        BF  11 00301          BRB     19$                               ; 1100
```

```
                    60           10  88 00303 24$:     BISB2   #16, (R0)                          ; 1119
                                 BA  11 00306          BRB     19$                                ; 1100
                    60           20  88 00308 25$:     BISB2   #32, (R0)                          ; 1124
                                 B5  11 0030B          BRB     19$                                ; 1093
                         00B4    C6  9F 0030D 26$:     PUSHAB  SUMMARY_DESC                       ; 1133
                                 57  DD 00311          PUSHL   R7                                 ;
                                 02  DD 00313          PUSHL   #2                                 ;
                     0008132C    8F  DD 00315          PUSHL   #529196                            ;
                    6A           04  FB 0031B          CALLS   #4, LIB$STOP                       ;
                                 A2  11 0031E          BRB     19$                                ; 1088
                    50     10    A5  D0 00320 27$:     MOVL    SUMMARY_FLAG, R0                   ; 1136
                                 60  D5 00324          TSTL    (R0)                               ;
                                 03  12 00326          BNEQ    28$                                ;
                    60           01  88 00328          BISB2   #1, (R0)                           ; 1137
                    50           65  D0 0032B 28$:     MOVL    OPTION_FLAG, R0                    ; 1145
          33        60           06  E0 0032E          BBS     #6, (R0), 29$                      ;
          2F        60           04  E0 00332          BBS     #4, (R0), 29$                      ; 1146
               01   A0     80    8F  88 00336          BISB2   #128, 1(R0)                        ; 1153
                    51     FC    A5  D0 0033B          MOVL    INCLUDE_MASK, R1                   ; 1154
               02   A1           10  8A 0033F          BICB2   #16, 2(R1)                         ;
                    50     E4    A5  D0 00343          MOVL    EXCLUDE_MASK, R0                   ; 1155
               02   A0           10  8A 00347          BICB2   #16, 2(R0)                         ;
               02   A1           20  8A 0034B          BICB2   #32, 2(R1)                         ; 1156
               02   A0           20  8A 0034F          BICB2   #32, 2(R0)                         ; 1157
               02   A1     40    8F  8A 00353          BICB2   #64, 2(R1)                         ; 1158
               02   A0     40    8F  8A 00358          BICB2   #64, 2(R0)                         ; 1159
               02   A1           08  8A 0035D          BICB2   #8, 2(R1)                          ; 1160
               02   A0           08  8A 00361          BICB2   #8, 2(R0)                          ; 1161
                           0C    A6  9F 00365 29$:     PUSHAB  FULL_DESC                          ; 1168
                    68           01  FB 00368          CALLS   #1, CLI$PRESENT                    ;
                    52           50  D0 0036B          MOVL    R0, STATUS                         ;
           00000000G 8F          52  D1 0036E          CMPL    STATUS, #CLI$_PRESENT              ; 1169
                                 0B  12 00375          BNEQ    30$                                ;
                                 53  D6 00377          INCL    I                                  ; 1175
                    50     04    A5  D0 00379          MOVL    PARSER_DATA, R0                    ; 1176
                    60           02  90 0037D          MOVB    #2, (R0)                           ;
                                 12  11 00380          BRB     31$                                ; 1169
           00000000G 8F          52  D1 00382 30$:     CMPL    STATUS, #CLI$_NEGATED              ; 1180
                                 09  12 00389          BNEQ    31$                                ;
                    50     04    A5  D0 0038B          MOVL    PARSER_DATA, R0                    ; 1186
                                 60  94 0038F          CLRB    (R0)                               ;
                    50           01  90 00391          MOVB    #1, FULL_NEGATE                    ; 1187
                           E0    A6  9F 00394 31$:     PUSHAB  BRIEF_DESC                         ; 1195
                    68           01  FB 00397          CALLS   #1, CLI$PRESENT                    ;
                    09           50  E9 0039A          BLBC    R0, 32$                            ;
                                 53  D6 0039D          INCL    I                                  ; 1201
                    50     04    A5  D0 0039F          MOVL    PARSER_DATA, R0                    ; 1202
                    60           01  90 003A3          MOVB    #1, (R0)                           ;
                           5C    A6  9F 003A6 32$:     PUSHAB  REGISTER_DUMP_DESC                 ; 1209
                    68           01  FB 003A9          CALLS   #1, CLI$PRESENT                    ;
                    20           50  E9 003AC          BLBC    R0, 34$                            ;
                                 53  D6 003AF          INCL    I                                  ; 1216
                    50     04    A5  D0 003B1          MOVL    PARSER_DATA, R0                    ; 1217
                    60           03  90 003B5          MOVB    #3, (R0)                           ;
                    50           65  D0 003B8          MOVL    OPTION_FLAG, R0                    ; 1219
          07        60           06  E1 003BB          BBC     #6, (R0), 33$                      ;
                    50     C4    A5  9E 003BF          MOVAB   DEV_ENTRY_KEY, R0                  ; 1220
```

```
                         09               50  E9 003C3         BLBC    R0, 34$
                            0000000G      8F  DD 003C6 33$:    PUSHL   #ERF_DEVSELREQ                  ; 1226
                         6A               01  FB 003CC         CALLS   #1, LIB$STOP                    ;
                         50         04    A5  D0 003CF 34$:    MOVL    PARSER_DATA, R0                 ; 1234
                                          60  95 003D3         TSTB    (R0)                            ;
                                          10  13 003D5         BEQL    35$                             ;
                         50               65  D0 003D7         MOVL    OPTION_FLAG, R0                 ; 1240
               09        60               01  E1 003DA         BBC     #1, (R0), 35$                   ;
                            000812E3      8F  DD 003DE         PUSHL   #529123                         ; 1246
                         6A               01  FB 003E4         CALLS   #1, LIB$STOP                    ;
                         01               53  D1 003E7 35$:    CMPL    I, #1                           ; 1252
                                          09  15 003EA         BLEQ    36$                             ;
                            000812E3      8F  DD 003EC         PUSHL   #529123                         ; 1257
                         6A               01  FB 003F2         CALLS   #1, LIB$STOP                    ;
                         50         FC    A5  D0 003F5 36$:    MOVL    INCLUDE_MASK, R0                ; 1265
                         51         FC    A7  D2 003F9         MCOML   KEYWRD_MASK, R1                 ;
               51        60               51  CB 003FD         BICL3   R1, (R0), R1                    ;
                         50         E4    A5  D0 00401         MOVL    EXCLUDE_MASK, R0                ; 1266
                         53         FC    A7  D2 00405         MCOML   KEYWRD_MASK, R3                 ;
               50        60               53  CB 00409         BICL3   R3, (R0), R0                    ;
                         52               51  D2 0040D         MCOML   R1, STATUS                      ;
               52        50               52  CB 00410         BICL3   STATUS, R0, STATUS              ;
                                          10  13 00414         BEQL    37$                             ; 1268
                                    1C    A6  9F 00416         PUSHAB  INCLUDE_DESC                    ; 1274
                                          56  DD 00419         PUSHL   R6                              ;
                                          02  DD 0041B         PUSHL   #2                              ;
                            0000000G      8F  DD 0041D         PUSHL   #ERF_CNFQUAVAL                  ;
                         6A               04  FB 00423         CALLS   #4, LIB$STOP                    ;
                 0000000V 00              00  FB 00426 37$:    CALLS   #0, CLASS_OPTION_CHECK          ; 1281
                         50               01  D0 0042D         MOVL    #1, R0                          ; 1287
                                              04 00430         RET                                     ; 1288

; Routine Size:  1073 bytes,    Routine Base:  $CODE + 0000

;  853          1289  1
```

```
;  855     1290  1 ROUTINE GET_DEVICE_SELECT (temp_desc) =         ! Device selection parsing
;  856     1291  1
;  857     1292  1 !++
;  858     1293  1 !
;  859     1294  1 ! Functional Description:
;  860     1295  1 !
;  861     1296  1 ! This routine determines if the 'value' specified with the /device or /exclude
;  862     1297  1 ! qualifier was a valid keyword and translates it to the device class
;  863     1298  1 ! designation.
;  864     1299  1 !
;  865     1300  1 ! Calling Sequence:
;  866     1301  1 !
;  867     1302  1 !    GET_DEVICE_SELECT (temp_desc)
;  868     1303  1 !
;  869     1304  1 ! Input Parameters:
;  870     1305  1 !
;  871     1306  1 !    Temp_desc = the 'value' associated with the qualifier
;  872     1307  1 !
;  873     1308  1 ! Output Parameters:
;  874     1309  1 !
;  875     1310  1 !    This routine will ****
;  876     1311  1 !
;  877     1312  1 !--
;  878     1313  2 Begin
;  879     1314  2
;  880     1315  2 LITERAL
;  881     1316  2     Max_keywords = 16 ;                   ! Maximum number of keywords
;  882     1317  2
;  883     1318  2 LOCAL
;  884     1319  2     Device_class,                         ! Device class storage
;  885     1320  2     Key_value,                            ! Key value storage
;  886     1321  2     Status ;                              ! Return status storage
;  887     1322  2
;  888     1323  2 !
;  889     1324  2 ! Create the device keyword table.
;  890     1325  2 !
;  891     1326  2 OWN
;  892     1327  2     !
;  893     1328  2     ! Define the device class and device entry keywords associated
;  894     1329  2     ! with the /exclude and /include qualifiers.
;  895     1330  2     !
;  896     1331  2     Dev_class_keywords:
;  897   P 1332  2         $LIB_KEY_TABLE (
;  898   P 1333  2 !           (Async_communications, 00),
;  899   P 1334  2             (Buses, 01),
;  900   P 1335  2             (Disks, 02),
;  901   P 1336  2             (Realtime, 03),
;  902   P 1337  2             (Sync_communications, 04),
;  903     1338  2             (Tapes, 05)),
;  904     1339  2
;  905     1340  2     Dev_entry_keywords:
;  906   P 1341  2         $LIB_KEY_TABLE (
;  907   P 1342  2             (Bugchecks, 06),
;  908   P 1343  2             (Control_entries, 07),
;  909   P 1344  2             (Cpu_entries, 08),
;  910   P 1345  2             (Device_errors, 9),
;  911   P 1346  2             (Machine_checks, 10),
```

```
                        (Memory, 11),
                        (Timeouts, 12),
                        (Volume_changes, 13),
                        (Attentions, 14),
                        (Unsolicited_mscp, 15),
                        (Unknown, 16)) ;

MAP
        Temp_desc:  REF $BBLOCK ;


!
! Allocate the necessary storage (zero filled) and initialize
! the device select queue entry.
!
Dev_select = GET_VM (q_entry_size) ;

Dev_select[dev$w_unit] = (-1) ;
Dev_select[dev$v_node_name_wild] = false ;
Dev_select[dev$v_exclude_flg] = false ;


!
! Determine if it is a device class keyword.
!
Dev_class_key = true ;
If NOT (status = LIB$LOOKUP_KEY(.temp_desc,dev_class_keywords,key_value))
Then
        !
        ! Not a device class keyword, determine if it is
        ! a device entry keyword.
        !
        Begin
        Dev_class_key = false ;
        Dev_entry_key = true ;
        If NOT (status = LIB$LOOKUP_KEY(.temp_desc,dev_entry_keywords,key_value))
        Then
                !
                ! Not a device entry keyword determine if it is a
                ! device specification.
                !
                Begin
                Dev_entry_key = false ;
                If NOT (PARSE_DEVNAME (.temp_desc))
                Then
                        !
                        ! Not a device specification either, return to calling routine.
                        !
                        Return false
                Else
                        !
                        ! Valid device specification, the name and
                        ! unit number are already stored in the queue entry.
                        !
                        ! Translate the device name to a device class.
                        !
                        Begin
```

```
;  969     1404  5            If NOT .wild_carded_device
;  970     1405  5            Then
;  971     1406  6                Begin
;  972     1407  6                If NOT TRANSLATE_DEVICE (dev_name,device_class)
;  973     1408  6                Then
;  974     1409  6                    !
;  975     1410  6                    ! Device not found, notify the user and exit.
;  976     1411  6                    !
;  977     1412  6                    Return false
;  978     1413  6                Else
;  979     1414  6                    !
;  980     1415  6                    ! Device found, save the device class in the device
;  981     1416  6                    ! select queue entry.
;  982     1417  6                    !
;  983     1418  6                    Dev_select[dev$b_class] = .device_class ;
;  984     1419  6
;  985     1420  6                !
;  986     1421  6                ! Search any entries already in the queue to ensure
;  987     1422  6                ! there are no conflicts between the selected
;  988     1423  6                ! device and any device class(es) already selected.
;  989     1424  6                !
;  990     1425  6                If NOT DEVICE_OPTION_CHECK ()
;  991     1426  6                Then
;  992     1427  6                    !
;  993     1428  6                    ! Like entry already in the queue.
;  994     1429  6                    ! (/include=MF,MF or /exclude=MF,MF)
;  995     1430  6                    !
;  996     1431  6                    Return true ;
;  997     1432  5                End ;
;  998     1433  5
;  999     1434  5            !
; 1000     1435  5            ! Insert entry in the queue. The LIB$INSQTI creates
; 1001     1436  5            ! a self relative queue that is interlocked.
; 1002     1437  5            !
; 1003     1438  6            If NOT (status = LIB$INSQTI (.dev_select,root_flink))
; 1004     1439  5            Then
; 1005     1440  5                !
; 1006     1441  5                ! The entry could not be placed in the queue, notify
; 1007     1442  5                ! the user and exit.
; 1008     1443  5                !
; 1009     1444  5                Signal_stop (.status)
; 1010     1445  5            Else
; 1011     1446  5                !
; 1012     1447  5                ! Entry was successfully entered in queue.
; 1013     1448  5                !
; 1014     1449  6                Begin
; 1015     1450  6                If .exclude_flag
; 1016     1451  6                Then
; 1017     1452  6                    Exclude_q_entry_cnt = .exclude_q_entry_cnt + 1
; 1018     1453  6                Else
; 1019     1454  6                    Include_q_entry_cnt = .include_q_entry_cnt + 1 ;
; 1020     1455  6
; 1021     1456  6                Que_entry_cnt = .exclude_q_entry_cnt + .include_q_entry_cnt ;
; 1022     1457  5                End ;
; 1023     1458  4            End ;
; 1024     1459  3        End ;
; 1025     1460  2    End ;
```

```
;  1026     1461  2
;  1027     1462  2 !
;  1028     1463  2 ! Valid keyword, set up the exclude and include option
;  1029     1464  2 ! selection indicators.
;  1030     1465  2 !
;  1031     1466  3 If (.dev_entry_key) OR (.dev_class_key)
;  1032     1467  2 Then
;  1033     1468  3     Begin
;  1034     1469  3     If .option_flag[opt$v_include_qual]
;  1035     1470  3     Then
;  1036     1471  4         Begin
;  1037     1472  4         Case .key_value from 1 to max_keywords of
;  1038     1473  4         Set
;  1039     1474  4 !        [0]:                           ! Asynchronous communications device class
;  1040     1475  4 !            Begin
;  1041     1476  4 !            Include_mask[inc$v_async_comm] = true ;
;  1042     1477  4 !            Include_class[async] = DC$_ACOM ;
;  1043     1478  4 !            Include_key[async] = .key_value ;
;  1044     1479  4 !            Include_mask[inc$v_dev_class_select] = true ;
;  1045     1480  4 !            End ;
;  1046     1481  4
;  104      1482  4             [1]:                           ! Bus device class
;  1048     1483  5                 Begin
;  1049     1484  5                 Include_mask[inc$v_buses] = true ;
;  1050     1485  5                 Include_class[bus] = DC$_BUS ;
;  1051     1486  5                 Include_key[bus] = .key_value ;
;  1052     1487  5                 Include_mask[inc$v_dev_class_select] = true ;
;  1053     1488  4                 End ;
;  1054     1489  4
;  1055     1490  4             [2]:                           ! Disk device class
;  1056     1491  5                 Begin
;  1057     1492  5                 Include_mask[inc$v_disks] = true ;
;  1058     1493  5                 Include_class[disk] = DC$_DISK ;
;  1059     1494  5                 Include_key[disk] = .key_value ;
;  1060     1495  5                 Include_mask[inc$v_dev_class_select] = true ;
;  1061     1496  4                 End ;
;  1062     1497  4
;  1063     1498  4             [3]:                           ! Realtime class
;  1064     1499  5                 Begin
;  1065     1500  5                 Include_mask[inc$v_realtime] = true ;
;  1066     1501  5                 Include_class[realtime] = DC$_REALTIME ;
;  1067     1502  5                 Include_key[realtime] = .key_value ;
;  1068     1503  5                 Include_mask[inc$v_dev_class_select] = true ;
;  1069     1504  4                 End ;
;  1070     1505  4
;  1071     1506  4             [4]:                           ! Synchronous communication device class
;  1072     1507  5                 Begin
;  1073     1508  5                 Include_mask[inc$v_sync_comm] = true ;
;  1074     1509  5                 Include_class[sync] = DC$_SCOM ;
;  1075     1510  5                 Include_key[sync] = .key_value ;
;  1076     1511  5                 Include_mask[inc$v_dev_class_select] = true ;
;  1077     1512  4                 End ;
;  1078     1513  4
;  1079     1514  4             [5]:                           ! Tapes device class
;  1080     1515  5                 Begin
;  1081     1516  5                 Include_mask[inc$v_tapes] = true ;
;  1082     1517  5                 Include_class[tape] = DC$_TAPE ;
```

```
;  1083          1518  5            Include_key[tape] = .key_value ;
;  1084          1519  5            Include_mask[inc$v_dev_class_select] = true ;
;  1085          1520  4            End ;
;  1086          1521  4
;  1087          1522  4        [6]:                                ! Bugcheck entries
;  1088          1523  5            Begin
;  1089          1524  5            Include_mask[inc$v_bugchks] = true ;
;  1090          1525  5            Include_mask[inc$v_entry_select] = true ;
;  1091          1526  5            !
;  1092          1527  5            ! Determine if a specific type of bugcheck entry
;  1093          1528  5            ! was selected.
;  1094          1529  5            !
;  1095          1530  5            !*****get value associated with bugchecks
;  1096          1531  4            End ;
;  1097          1532  4
;  1098          1533  4        [7]:                                ! Control entries
;  1099          1534  5            Begin
;  1100          1535  5            Include_mask[inc$v_control_entry] = true ;
;  1101          1536  5            Include_mask[inc$v_entry_select] = true ;
;  1102          1537  4            End ;
;  1103          1538  4
;  1104          1539  4        [8]:                                ! Cpu entries
;  1105          1540  5            Begin
;  1106          1541  5            Include_mask[inc$v_cpu_entry] = true ;
;  1107          1542  5            Include_mask[inc$v_entry_select] = true ;
;  1108          1543  4            End ;
;  1109          1544  4
;  1110          1545  4        [9]:                                ! Device error entries
;  1111          1546  5            Begin
;  1112          1547  5            Include_mask[inc$v_dev_errors] = true ;
;  1113          1548  5            Include_mask[inc$v_entry_select] = true ;
;  1114          1549  4            End ;
;  1115          1550  4
;  1116          1551  4        [10]:                               ! Machine check entries
;  1117          1552  5            Begin
;  1118          1553  5            Include_mask[inc$v_machine_chks] = true ;
;  1119          1554  5            Include_mask[inc$v_entry_select] = true ;
;  1120          1555  4            End ;
;  1121          1556  4
;  1122          1557  4        [11]:                               ! Memory entries
;  1123          1558  5            Begin
;  1124          1559  5            Include_mask[inc$v_memory] = true ;
;  1125          1560  5            Include_mask[inc$v_entry_select] = true ;
;  1126          1561  4            End ;
;  1127          1562  4
;  1128          1563  4        [12]:                               ! Device timeout entries
;  1129          1564  5            Begin
;  1130          1565  5            Include_mask[inc$v_dev_timeouts] = true ;
;  1131          1566  5            Include_mask[inc$v_entry_select] = true ;
;  1132          1567  4            End ;
;  1133          1568  4
;  1134          1569  4        [13]:                               ! Volume change entries
;  1135          1570  5            Begin
;  1136          1571  5            Include_mask[inc$v_volume] = true ;
;  1137          1572  5            Include_mask[inc$v_entry_select] = true ;
;  1138          1573  4            End ;
;  1139          1574  4
```

```
;  1140          1575  4          [14]:                               ! Device attention entries
;  1141          1576  5              Begin
;  1142          1577  5              Include_mask[inc$v_dev_attentions] = true ;
;  1143          1578  5              Include_mask[inc$v_entry_select] = true ;
;  1144          1579  4              End ;
;  1145          1580  4
;  1146          1581  4          [15]:                               ! Unsolicted mscp entries (logmscp)
;  1147          1582  5              Begin
;  1148          1583  5              Include_mask[inc$v_unsol_mscp] = true ;
;  1149          1584  5              Include_mask[inc$v_entry_select] = true ;
;  1150          1585  4              End ;
;  1151          1586  4
;  1152          1587  4          [16]:                               ! Unknown entry
;  1153          1588  5              Begin
;  1154          1589  5              Include_mask[inc$v_unknown_entry] = true ;
;  1155          1590  5              Include_mask[inc$v_entry_select] = true ;
;  1156          1591  4              End ;
;  1157          1592  4
;  1158          1593  4          [Outrange]:
;  1159          1594  5              Begin
;  1160          1595  5              Return false ;
;  1161          1596  4              End ;
;  1162          1597  4
;  1163          1598  4          TES ;
;  1164          1599  3          End ;
;  1165          1600  3
;  1166          1601  4          If (.option_flag[opt$v_exclude_qual] AND .exclude_flag)
;  1167          1602  3          Then
;  1168          1603  3              !
;  1169          1604  3              ! Set up the /exclude option selection indicators.
;  1170          1605  3              !
;  1171          1606  4              Begin
;  1172          1607  4              Case .key_value from 1 to max_keywords of
;  1173          1608  4              Set
;  1174          1609  4  !           [0]:                                ! Asynchronous communications device class
;  1175          1610  4  !               Begin
;  1176          1611  4  !               Exclude_mask[exc$v_async_comm] = true ;
;  1177          1612  4  !               Exclude_class[async] = DC$_ACOM ;
;  1178          1613  4  !               Exclude_key[async] = exc$v_async_comm ;
;  1179          1614  4  !               Exclude_mask[exc$v_dev_class_select] = true ;
;  1180          1615  4  !               End ;
;  1181          1616  4
;  1182          1617  4              [1]:                                ! Bus device class
;  1183          1618  5                  Begin
;  1184          1619  5                  Exclude_mask[exc$v_buses] = true ;
;  1185          1620  5                  Exclude_class[bus] = DC$_BUS ;
;  1186          1621  5                  Exclude_key[bus] = .key_value ;
;  1187          1622  5                  Exclude_mask[exc$v_dev_class_select] = true ;
;  1188          1623  4                  End ;
;  1189          1624  4
;  1190          1625  4              [2]:                                ! Disk device class
;  1191          1626  5                  Begin
;  1192          1627  5                  Exclude_mask[exc$v_disks] = true ;
;  1193          1628  5                  Exclude_class[disk] = DC$_DISK ;
;  1194          1629  5                  Exclude_key[disk] = .key_value ;
;  1195          1630  5                  Exclude_mask[exc$v_dev_class_select] = true ;
;  1196          1631  4                  End ;
```

```
;  1197     1632  4
;  1198     1633  4            [3]:                                    ! Realtime device class
;  1199     1634  5                Begin
;  1200     1635  5                Exclude_mask[exc$v_realtime] = true ;
;  1201     1636  5                Exclude_class[realtime] = DC$_REALTIME ;
;  1202     1637  5                Exclude_key[realtime] = .key_value ;
;  1203     1638  5                Exclude_mask[exc$v_dev_class_select] = true ;
;  1204     1639  4                End ;
;  1205     1640  4
;  1206     1641  4            [4]:                                    ! Synchronous communication device class
;  1207     1642  5                Begin
;  1208     1643  5                Exclude_mask[exc$v_sync_comm] = true ;
;  1209     1644  5                Exclude_class[sync] = DC$_SCOM ;
;  1210     1645  5                Exclude_key[sync] = .key_value ;
;  1211     1646  5                Exclude_mask[exc$v_dev_class_select] = true ;
;  1212     1647  4                End ;
;  1213     1648  4
;  1214     1649  4            [5]:                                    ! Tape device class
;  1215     1650  5                Begin
;  1216     1651  5                Exclude_mask[exc$v_tapes] = true ;
;  1217     1652  5                Exclude_class[tape] = DC$_TAPE ;
;  1218     1653  5                Exclude_key[tape] = .key_value ;
;  1219     1654  5                Exclude_mask[exc$v_dev_class_select] = true ;
;  1220     1655  4                End ;
;  1221     1656  4
;  1222     1657  4            [6]:                                    ! Bugcheck entries
;  1223     1658  5                Begin
;  1224     1659  5                Exclude_mask[exc$v_bugchks] = true ;
;  1225     1660  5                Exclude_mask[exc$v_entry_select] = true ;
;  1226     1661  5                !
;  1227     1662  5                ! Determine if a specific type of bugcheck entry
;  1228     1663  5                ! was selected.
;  1229     1664  5                !
;  1230     1665  5                !***** Get value associated with it
;  1231     1666  4                End ;
;  1232     1667  4
;  1233     1668  4            [7]:                                    ! Control entries
;  1234     1669  5                Begin
;  1235     1670  5                Exclude_mask[exc$v_control_entry] = true ;
;  1236     1671  5                Exclude_mask[exc$v_entry_select] = true ;
;  1237     1672  4                End ;
;  1238     1673  4
;  1239     1674  4            [8]:                                    ! Cpu entries
;  1240     1675  5                Begin
;  1241     1676  5                Exclude_mask[exc$v_cpu_entry] = true ;
;  1242     1677  5                Exclude_mask[exc$v_entry_select] = true ;
;  1243     1678  4                End ;
;  1244     1679  4
;  1245     1680  4            [9]:                                    ! Device error entries
;  1246     1681  5                Begin
;  1247     1682  5                Exclude_mask[exc$v_dev_errors] = true ;
;  1248     1683  5                Exclude_mask[exc$v_entry_select] = true ;
;  1249     1684  4                End ;
;  1250     1685  4
;  1251     1686  4            [10.:                                   ! Machine check entries
;  1252     1687  5                Begin
;  1253     1688  5                Exclude_mask[exc$v_machine_chks] = true ;
```

```
;  1254     1689  5              Exclude_mask[exc$v_entry_select] = true ;
;  1255     1690  4              End ;
;  1256     1691  4
;  1257     1692  4          [11]:                                  ! Memory entries
;  1258     1693  5              Begin
;  1259     1694  5              Exclude_mask[exc$v_memory] = true ;
;  1260     1695  5              Exclude_mask[exc$v_entry_select] = true ;
;  1261     1696  4              End ;
;  1262     1697  4
;  1263     1698  4          [12]:                                  ! Device timeout entries
;  1264     1699  5              Begin
;  1265     1700  5              Exclude_mask[exc$v_dev_timeouts] = true ;
;  1266     1701  5              Exclude_mask[exc$v_entry_select] = true ;
;  1267     1702  4              End ;
;  1268     1703  4
;  1269     1704  4          [13]:                                  ! Volume entries (mount/dismount)
;  1270     1705  5              Begin
;  1271     1706  5              Exclude_mask[exc$v_volume] = true ;
;  1272     1707  5              Exclude_mask[exc$v_entry_select] = true ;
;  1273     1708  4              End ;
;  1274     1709  4
;  1275     1710  4          [14]:                                  ! Device attention entries
;  1276     1711  5              Begin
;  1277     1712  5              Exclude_mask[exc$v_dev_attentions] = true ;
;  1278     1713  5              Exclude_mask[exc$v_entry_select] = true ;
;  1279     1714  4              End ;
;  1280     1715  4
;  1281     1716  4          [15]:                                  ! Unsolicited mscp entries (logmscp)
;  1282     1717  5              Begin
;  1283     1718  5              Exclude_mask[exc$v_unsol_mscp] = true ;
;  1284     1719  5              Exclude_mask[exc$v_entry_select] = true ;
;  1285     1720  4              End ;
;  1286     1721  4
;  1287     1722  4          [16]:                                  ! Unnknown entry
;  1288     1723  5              Begin
;  1289     1724  5              Exclude_mask[exc$v_unknown_entry] = true ;
;  1290     1725  5              Exclude_mask[exc$v_entry_select] = true ;
;  1291     1726  4              End ;
;  1292     1727  4
;  1293     1728  4          [Outrange]:
;  1294     1729  5              Begin
;  1295     1730  5              Return false ;
;  1296     1731  4              End ;
;  1297     1732  4          TES ;
;  1298     1733  3          End ;
;  1299     1734  2      End ;
;  1300     1735  2
;  1301     1736  2 !
;  1302     1737  2 ! Output data is set up.
;  1303     1738  2 !
;  1304     1739  2 Return true ;
;  1305     1740  2
;  1306     1741  1 End ;              ! Routine


                                                    .PSECT  $PLIT,NOWRT,NOEXE,  PIC,2
```

```
                                                00 00 53 45 53 55 42 05  00184 P.ABV:   .ASCII  <5>\BUSES\<0><0>
                                                00 00 53 4B 53 49 44 05  00188 P.ABW:   .ASCII  <5>\DISKS\<0><0>
                                  00 00 00 45   4D 49 54 4C 41 45 52 08  00194 P.ABX:   .ASCII  <8>\REALTIME\<0><0><0>
41 43 49 4E 55 4D 4D 4F 43 5F 43 4E 59 53 13  001A0 P.ABY:   .ASCII  <19>\SYNC_COMMUNICATIONS\
                                                53 4E 4F 49 54        001AF
                                                00 00 53 45 50 41 54 05  001B4 P.ABZ:   .ASCII  <5>\TAPES\<0><0>
                                  00 00 53 4B   43 45 48 43 47 55 42 09  001BC P.ACA:   .ASCII  <9>\BUGCHECKS\<0><0>
45 49 52 54 4E 45 5F 4C 4F 52 54 4E 4F 43 0F  001C8 P.ACB:   .ASCII  <15>\CONTROL_ENTRIES\
                                                53                    001D7
                                  53 45 49 52   54 4E 45 5F 55 50 43 0B  001D8 P.ACC:   .ASCII  <11>\CPU_ENTRIES\
00 53 52 4F 52 52 45 5F 45 43 49 56 45 44 0D  001E4 P.ACD:   .ASCII  <13>\DEVICE_ERRORS\<0><0>
                                                00                    001F3
53 4B 43 45 48 43 5F 45 4E 49 48 43 41 4D 0E  001F4 P.ACE:   .ASCII  <14>\MACHINE_CHECKS\<0>
                                                00                    00203
                                                00 59 52 4F 4D 45 4D 06  00204 P.ACF:   .ASCII  <6>\MEMORY\<0>
                                  00 00 00 53   54 55 4F 45 4D 49 54 08  0020C P.ACG:   .ASCII  <8>\TIMEOUTS\<0><0><0>
53 45 47 4E 41 48 43 5F 45 4D 55 4C 4F 56 0E  00218 P.ACH:   .ASCII  <14>\VOLUME_CHANGES\<0>
                                                00                    00227
                                  00 53 4E 4F   49 54 4E 45 54 54 41 0A  00228 P.ACI:   .ASCII  <10>\ATTENTIONS\<0>
53 4D 5F 44 45 54 49 43 49 4C 4F 4E 55 10  00234 P.ACJ:   .ASCII  <16>\UNSOLICITED_MSCP\<0><0><0>
                                                00 00 00 50 43        00243
                                                4E 57 4F 4E 4B 4E 55 07  00248 P.ACK:   .ASCII  <7>\UNKNOWN\

                                                                      .PSECT  $OWN$,NOEXE,  PIC,2

                                                          0000000A  00038 DEV_CLASS_KEYWORDS:
                                                                      .LONG   10
                                                          00000000' 0003C         .ADDRESS P.ABV
                                                          00000001  00040         .LONG   1
                                                          00000000' 00044         .ADDRESS P.ABW
                                                          00000002  00048         .LONG   2
                                                          00000000' 0004C         .ADDRESS P.ABX
                                                          00000003  00050         .LONG   3
                                                          00000000' 00054         .ADDRESS P.ABY
                                                          00000004  00058         .LONG   4
                                                          00000000' 0005C         .ADDRESS P.ABZ
                                                          00000005  00060         .LONG   5
                                                          00000016  00064 DEV_ENTRY_KEYWORDS:
                                                                      .LONG   22
                                                          00000000' 00068         .ADDRESS P.ACA
                                                          00000006  0006C         .LONG   6
                                                          00000000' 00070         .ADDRESS P.ACB
                                                          00000007  00074         .LONG   7
                                                          00000000' 00078         .ADDRESS P.ACC
                                                          00000008  0007C         .LONG   8
                                                          00000000' 00080         .ADDRESS P.ACD
                                                          00000009  00084         .LONG   9
                                                          00000000' 00088         .ADDRESS P.ACE
                                                          0000000A  0008C         .LONG   10
                                                          00000000' 00090         .ADDRESS P.ACF
                                                          0000000B  00094         .LONG   11
                                                          00000000' 00098         .ADDRESS P.ACG
                                                          0000000C  0009C         .LONG   12
                                                          00000000' 000A0         .ADDRESS P.ACH
                                                          0000000D  000A4         .LONG   13
                                                          00000000' 000A8         .ADDRESS P.ACI
```

```
                        0000000E  000AC          .LONG   14                                      ;
                        00000000' 000B0          .ADDRESS P.ACJ                                  ;
                        0000000F  000B4          .LONG   15                                      ;
                        00000000' 000B8          .ADDRESS P.ACK                                  ;
                        00000010  000BC          .LONG   16                                      ;


                                                 .PSECT  $CODE,NOWRT,  PIC,2

                                001C 00000 GET_DEVICE_SELECT:
                                           .WORD   Save R2,R3,R4                                ; 1290
                              54 00000000G  00  9E 00002 MOVAB   LIB$LOOKUP_KEY, R4             ;
                              53 00000000'  00  9E 00009 MOVAB   INCLUDE_MASK, R3               ;
                              5E            08  C2 00010 SUBL2   #8, SP                       ;
                                            20  DD 00013 PUSHL   #32                            ; 1362
                   00000000V  00            01  FB 00015 CALLS   #1, GET_VM                   ;
                          D0  A3            50  D0 0001C MOVL    R0, DEV_SELECT              ;
                          1B  A0            01  AE 00020 MNEGW   #1, 27(R0)                  ; 1364
                          1E  A0            03  8A 00024 BICB2   #3, 30(R0)                  ; 1366
                          C4  A3            01  D0 00028 MOVL    #1, DEV_CLASS_KEY           ; 1372
                                            5E  DD 0002C PUSHL   SP                          ; 1373
                                 00000000'  00  9F 0002E PUSHAB  DEV_CLASS_KEYWORDS          ;
                                       04   AC  DD 00034 PUSHL   TEMP_DESC                   ;
                                 64         03  FB 00037 CALLS   #3, LIB$LOOKUP_KEY          ;
                                 52         50  D0 0003A MOVL    R0, STATUS                  ;
                                 75         52  E8 0003D BLBS    STATUS, 5$                  ;
                                       C4   A3  D4 00040 CLRL    DEV_CLASS_KEY               ; 1380
                                 C8    A3         01  D0 00043 MOVL    #1, DEV_ENTRY_KEY       ; 1381
                                            5E  DD 00047 PUSHL   SP                          ; 1382
                                 00000000'  00  9F 00049 PUSHAB  DEV_ENTRY_KEYWORDS          ;
                                       04   AC  DD 0004F PUSHL   TEMP_DESC                   ;
                                 64         03  FB 00052 CALLS   #3, LIB$LOOKUP_KEY          ;
                                 52         50  D0 00055 MOVL    R0, STATUS                  ;
                                 75         52  E8 00058 BLBS    STATUS, 9$                  ;
                                       C8   A3  D4 0005B CLRL    DEV_ENTRY_KEY               ; 1389
                                       04   AC  DD 0005E PUSHL   TEMP_DESC                   ; 1390
                   00000000V  00            01  FB 00061 CALLS   #1, PARSE_DEVNAME          ;
                              11            50  E9 00068 BLBC    R0, 1$                      ;
                              29       1C   A3  E8 0006B BLBS    WILD_CARDED_DEVICE, 4$      ; 1404
                                       04   AE  9F 0006F PUSHAB  DEVICE_CLASS                ; 1407
                                       CC   A3  9F 00072 PUSHAB  DEV_NAME                   ;
                   00000000V  00            02  FB 00075 CALLS   #2, TRANSLATE_DEVICE       ;
                              03            50  E8 0007C 1$:    BLBS    R0, 2$              ;
                                      0256  31  0007F BRW     54$                             ;
                              50       D0   A3  D0 00082 2$:    MOVL    DEV_SELECT, R0      ; 1418
                            1D    A0       04  AE  90 00086 MOVB    DEVICE_CLASS, 29(R0)        ;
                   00000000V  00            00  FB 0008B CALLS   #0, DEVICE_OPTION_CHECK     ; 1425
                              03            50  E8 00092 BLBS    R0, 4$                      ;
                                      023C  31  00095 3$:    BRW     53$                     ;
                                 0000'      CF  9F 00098 4$:    PUSHAB  ROOT_FLINK          ; 1438
                                       D0   A3  DD 0009C PUSHL   DEV_SELECT                  ;
                   00000000G  00            02  FB 0009F CALLS   #2, LIB$INSQTI            ;
                              52            50  D0 0CJA6 MOVL    R0, STATUS                  ;
                              0B            52  E8 000A9 BLBS    STATUS, 6$                  ;
                                            52  DD 000AC PUSHL   STATUS                      ; 1444
                   00000000G  00            01  FB 000AE CALLS   #1, LIB$STOP              ;
```

```
                                    19  11 000B5 5$:     BRB     9$                            ;
                  05         DE     A3  E9 000B7 6$:     BLBC    EXCLUDE_FLAG, 7$              ; 1450
                             EC     A3  96 000BB         INCB    EXCLUDE_Q_ENTRY_CNT           ; 1452
                                    03  11 000BE         BRB     8$                            ;
                             ED     A3  96 000C0 7$:     INCB    INCLUDE_Q_ENTRY_CNT           ; 1454
                  50         EC     A3  9A 000C3 8$:     MOVZBL  EXCLUDE_Q_ENTRY_CNT, R0       ; 1456
                  51         ED     A3  9A 000C7         MOVZBL  INCLUDE_Q_ENTRY_CNT, R1       ;
        10    A3  50                51  A1 000CB         ADDW3   R1, R0, QUE_ENTRY_CNT         ;
                  04         C8     A3  E8 000D0 9$:     BLBS    DEV_ENTRY_KEY, 10$            ; 1466
                  BD         C4     A3  E9 000D4         BLBC    DEV_CLASS_KEY, 3$             ;
                  52         04     A3  D0 000D8 10$:    MOVL    OPTION_FLAG, R2               ; 1469
              03  62                06  E0 000DC         BBS     #6, (R2), 11$                 ;
                                  00EB  31 000E0         BRW     31$                           ;
                  50                6E  D0 000E3 11$:    MOVL    KEY_VALUE, R0                 ; 1472
              0F  01                50  CF 000E6         CASEL   R0, #1, #15                   ;
  0056      0044  0034            0023     000EA 12$:    .WORD   13$-12$,-                     ;
  008E      0085  007C            0067     000F2                 14$-12$,-                     ;
  00B4      00AA  00A0            0097     000FA                 15$-12$,-                     ;
  00D8      00CF  00C6            00BD     00102                 16$-12$,-                     ;
                                                                 17$-12$,-                     ;
                                                                 19$-12$,-                     ;
                                                                 20$-12$,-                     ;
                                                                 21$-12$,-                     ;
                                                                 22$-12$,-                     ;
                                                                 23$-12$,-                     ;
                                                                 24$-12$,-                     ;
                                                                 25$-12$,-                     ;
                                                                 26$-12$,-                     ;
                                                                 27$-12$,-                     ;
                                                                 28$-12$,-                     ;
                                                                 29$-12$                       ;
                                  01CB  31 0010A         BRW     54$                           ; 1595
                  51                63  D0 0010D 13$:    MOVL    INCLUDE_MASK, R1              ; 1484
                  61                02  88 00110         BISB2   #2, (R1)                      ;
              F1  A3         80     8F  90 00113         MOVB    #-128, INCLUDE_CLASS+1        ; 1485
              F9  A3                50  90 00118         MOVB    R0, INCLUDE_KEY+1             ; 1486
                                    42  11 0011C         BRB     18$                           ; 1487
                  51                63  D0 0011E 14$:    MOVL    INCLUDE_MASK, R1              ; 1492
                  61                04  88 00121         BISB2   #4, (R1)                      ;
              F2  A3                01  90 00124         MOVB    #1, INCLUDE_CLASS+2           ; 1493
              FA  A3                50  90 00128         MOVB    R0, INCLUDE_KEY+2             ; 1494
                                    32  11 0012C         BRB     18$                           ; 1495
                  51                63  D0 0012E 15$:    MOVL    INCLUDE_MASK, R1              ; 1500
                  61         40     8F  88 00131         BISB2   #64, (R1)                     ;
              F3  A3         60     8F  90 00135         MOVB    #96, INCLUDE_CLASS+3          ; 1501
              FB  A3                50  90 0013A         MOVB    R0, INCLUDE_KEY+3             ; 1502
                                    20  11 0013E         BRB     18$                           ; 1503
                  51                63  D0 00140 16$:    MOVL    INCLUDE_MASK, R1              ; 1508
                  61         80     8F  88 00143         BISB2   #128, (R1)                    ;
              F4  A3                20  90 00147         MOVB    #32, INCLUDE_CLASS+4          ; 1509
              FC  A3                50  90 0014B         MOVB    R0, INCLUDE_KEY+4             ; 1510
                                    0F  11 0014F         BRB     18$                           ; 1511
                  51                63  D0 00151 17$:    MOVL    INCLUDE_MASK, R1              ; 1516
              01  A1                01  88 00154         BISB2   #1, 1(R1)                     ;
              F5  A3                02  90 00158         MOVB    #2, INCLUDE_CLASS+5           ; 1517
              FD  A3                50  90 0015C         MOVB    R0, INCLUDE_KEY+5             ; 1518
              02  A1                20  88 00160 18$:    BISB2   #32, 2(R1)                    ; 1519
```

```
                                 68  11 00164       BRB     31$                        ; 1472
                       50        63  D0 00166 19$:  MOVL    INCLUDE_MASK, R0           ; 1524
                  01   A0        04  88 00169       BISB2   #4, 1(R0)
                                 5A  11 0016D       BRB     30$                        ; 1525
                       50        63  D0 0016F 20$:  MOVL    INCLUDE_MASK, R0           ; 1535
                  01   A0        08  88 00172       BISB2   #8, 1(R0)
                                 51  11 00176       BRB     30$                        ; 1536
                       50        63  D0 00178 21$:  MOVL    INCLUDE_MASK, R0           ; 1541
                  01   A0        10  88 0017B       BISB2   #16, 1(R0)
                                 48  11 0017F       BRB     30$                        ; 1542
                       50        63  D0 00181 22$:  MOVL    INCLUDE_MASK, R0           ; 1547
                  01   A0        20  88 00184       BISB2   #32, 1(R0)
                                 3F  11 00188       BRB     30$                        ; 1548
                       50        63  D0 0018A 23$:  MOVL    INCLUDE_MASK, R0           ; 1553
                  01   A0    40  8F  88 0018D       BISB2   #64, 1(R0)
                                 35  11 00192       BRB     30$                        ; 1554
                       50        63  D0 00194 24$:  MOVL    INCLUDE_MASK, R0           ; 1559
                  01   A0    80  8F  88 00197       BISB2   #128, 1(R0)
                                 2B  11 0019C       BRB     30$                        ; 1560
                       50        63  D0 0019E 25$:  MOVL    INCLUDE_MASK, R0           ; 1565
                  02   A0        01  88 001A1       BISB2   #1, 2(R0)
                                 22  11 001A5       BRB     30$                        ; 1566
                       50        63  D0 001A7 26$:  MOVL    INCLUDE_MASK, R0           ; 1571
                  02   A0        04  88 001AA       BISB2   #4, 2(R0)
                                 19  11 001AE       BRB     30$                        ; 1572
                       50        63  D0 001B0 27$:  MOVL    INCLUDE_MASK, R0           ; 1577
                  01   A0        02  88 001B3       BISB2   #2, 1(R0)
                                 10  11 001B7       BRB     30$                        ; 1578
                       50        63  D0 001B9 28$:  MOVL    INCLUDE_MASK, R0           ; 1583
                  02   A0        02  88 001BC       BISB2   #2, 2(R0)
                                 07  11 001C0       BRB     30$                        ; 1584
                       50        63  D0 001C2 29$:  MOVL    INCLUDE_MASK, R0           ; 1589
                  02   A0        08  88 001C5       BISB2   #8, 2(R0)
                  02   A0    40  8F  88 001C9 30$:  BISB2   #64, 2(R0)                 ; 1590
             03        62        04  E0 001CE 31$:  BBS     #4, (R2), 33$              ; 1601
                               00FF  31 001D2 32$:  BRW     53$                        ;
                       F9    DE  A3  E9 001D5 33$:  BLBC    EXCLUDE_FLAG, 32$          ;
                       50        6E  D0 001D9       MOVL    KEY_VALUE, R0              ; 1607
             0F        01        50  CF 001DC       CASEL   R0, #1, #15                ;
   0059    0046      0035      0023     001E0 34$:  .WORD   35$-34$,-                  ;
   0095    008B      0081      006B     001E8               36$-34$,-                  ;
   00BF    00B4      00A9      009F     001F0               37$-34$,-                  ;
   00E7    00DD      00D3      00C9     001F8               38$-34$,-                  ;
                                                            39$-34$,-                  ;
                                                            41$-34$,-                  ;
                                                            42$-34$,-                  ;
                                                            43$-34$,-                  ;
                                                            44$-34$,-                  ;
                                                            45$-34$,-                  ;
                                                            46$-34$,-                  ;
                                                            47$-34$,-                  ;
                                                            48$-34$,-                  ;
                                                            49$-34$,-                  ;
                                                            50$-34$,-                  ;
                                                            51$-34$                    ;
                               00D5  31 00200       BRW     54$                        ; 1730
                       51    E8  A3  D0 00203 35$:  MOVL    EXCLUDE_MASK, R1           ; 1619
```

```
         61        02 88 00207        BISB2  #2, (R1)                     ;
   D9    A3   80   8F 90 0020A        MOVB   #-128, EXCLUDE_CLASS+1       ; 1620
   E1    A3        50 90 0020F        MOVB   R0, EXCLUDE_KEY+1            ; 1621
                   46 11 00213        BRB    40$                          ; 1622
         51   E8   A3 D0 00215 36$:   MOVL   EXCLUDE_MASK, R1             ; 1627
         61        04 88 00219        BISB2  #4, (R1)                     ;
   DA    A3        01 90 0021C        MOVB   #1, EXCLUDE_CLASS+2          ; 1628
   E2    A3        50 90 00220        MOVB   R0, EXCLUDE_KEY+2            ; 1629
                   35 11 00224        BRB    40$                          ; 1630
         51   E8   A3 D0 00226 37$:   MOVL   EXCLUDE_MASK, R1             ; 1635
         61   40   8F 88 0022A        BISB2  #64, (R1)                    ;
   DB    A3   60   8F 90 0022E        MOVB   #96, EXCLUDE_CLASS+3         ; 1636
   E3    A3        50 90 00233        MOVB   R0, EXCLUDE_KEY+3            ; 1637
                   22 11 00237        BRB    40$                          ; 1638
         51   E8   A3 D0 00239 38$:   MOVL   EXCLUDE_MASK, R1             ; 1643
         61   80   8F 88 0023D        BISB2  #128, (R1)                   ;
   DC    A3        20 90 00241        MOVB   #32, EXCLUDE_CLASS+4         ; 1644
   E4    A3        50 90 00245        MOVB   R0, EXCLUDE_KEY+4            ; 1645
                   10 11 00249        BRB    40$                          ; 1646
         51   E8   A3 D0 0024B 39$:   MOVL   EXCLUDE_MASK, R1             ; 1651
   01    A1        01 88 0024F        BISB2  #1, 1(R1)                    ;
   DD    A3        02 90 00253        MOVB   #2, EXCLUDE_CLASS+5          ; 1652
   E5    A3        50 90 00257        MOVB   R0, EXCLUDE_KEY+5            ; 1653
   02    A1        20 88 0025B 40$:   BISB2  #32, 2(R1)                   ; 1654
                   73 11 0025F        BRB    53$                          ; 1607
         50   E8   A3 D0 00261 41$:   MOVL   EXCLUDE_MASK, R0             ; 1659
   01    A0        04 88 00265        BISB2  #4, 1(R0)                    ;
                   64 11 00269        BRB    52$                          ; 1660
         50   E8   A3 D0 0026B 42$:   MOVL   EXCLUDE_MASK, R0             ; 1670
   01    A0        08 88 0026F        BISB2  #8, 1(R0)                    ;
                   5A 11 00273        BRB    52$                          ; 1671
         50   E8   A3 D0 00275 43$:   MOVL   EXCLUDE_MASK, R0             ; 1676
   01    A0        10 88 00279        BISB2  #16, 1(R0)                   ;
                   50 11 0027D        BRB    52$                          ; 1677
         50   E8   A3 D0 0027F 44$:   MOVL   EXCLUDE_MASK, R0             ; 1682
   01    A0        20 88 00283        BISB2  #32, 1(R0)                   ;
                   46 11 00287        BRB    52$                          ; 1683
         50   E8   A3 D0 00289 45$:   MOVL   EXCLUDE_MASK, R0             ; 1688
   01    A0   40   8F 88 0028D        BISB2  #64, 1(R0)                   ;
                   3B 11 00292        BRB    52$                          ; 1689
         50   E8   A3 D0 00294 46$:   MOVL   EXCLUDE_MASK, R0             ; 1694
   01    A0   80   8F 88 00298        BISB2  #128, 1(R0)                  ;
                   30 11 0029D        BRB    52$                          ; 1695
         50   E8   A3 D0 0029F 47$:   MOVL   EXCLUDE_MASK, R0             ; 1700
   02    A0        01 88 002A3        BISB2  #1, 2(R0)                    ;
                   26 11 002A7        BRB    52$                          ; 1701
         50   E8   A3 D0 002A9 48$:   MOVL   EXCLUDE_MASK, R0             ; 1706
   02    A0        04 88 002AD        BISB2  #4, 2(R0)                    ;
                   1C 11 002B1        BRB    52$                          ; 1707
         50   E8   A3 D0 002B3 49$:   MOVL   EXCLUDE_MASK, R0             ; 1712
   01    A0        02 88 002B7        BISB2  #2, 1(R0)                    ;
                   12 11 002BB        BRB    52$                          ; 1713
         50   E8   A3 D0 002BD 50$:   MOVL   EXCLUDE_MASK, R0             ; 1718
   02    A0        02 88 002C1        BISB2  #2, 2(R0)                    ;
                   08 11 002C5        BRB    52$                          ; 1719
         50   E8   A3 D0 002C7 51$:   MOVL   EXCLUDE_MASK, R0             ; 1724
   02    A0        08 88 002CB        BISB2  #8, 2(R0)                    ;
```

```
                     02  A0      40  8F  88 002CF 52$:    BISB2   #64, 2(R0)                    ; 1725
                         50          01  D0 002D4 53$:    MOVL    #1, R0                        ; 1739
                                     04  002D7           RET                                   ;
                                     50  D4 002D8 54$:    CLRL    R0                            ; 1741
                                         04 002DA         RET                                   ;

; Routine Size:  731 bytes,    Routine Base:  $CODE + 0431

; 1307          1742  1
```

```
1 GLOBAL ROUTINE PARSE_DEVNAME (name) =
1
1 !++
1 !
1 ! Functional Description:
1 !
1 !     This routine parses a device name specification. It will validate the
1 !     length and range of the unit number and that the characters specified
1 !     as the name/controller are within a valid range. It will also check
1 !     for wildcard name and unit selection.
1 !
1 ! Calling Sequence:
1 !
1 !     PARSE_DEVNAME (device name descriptor)
1 !
1 ! Input Parameters:
1 !
1 !     Device name descriptor = device name passed by descriptor
1 !
1 ! Output Parameters:
1 !
1 !     *
1 !
1 !--
2 Begin
2
2 EXTERNAL
2     Include_desc ;
2
2 LITERAL
2     Min_len = 2,
2     Max_len = 14,
2     Unit_number_len = 5 ;
2
2 MAP
2     Dev_select: REF $BBLOCK,
2     Name:       REF $BBLOCK ;
2
2 LOCAL
2     I,                            ! Temporary counter
2     Name_size,                    ! Storage for node name size
2     Name_len:   Initial (0),      ! Storage for device name size
2     Ptr,                          ! Character string pointer
2     Ptr_unit,                     ! Character string pointer to unit number
2     Sp_chr_len: Initial (0),
2     Status,                       ! Completion status indicator
2     Str_len,                      ! String length
2     Tmp_ptr,                      ! Temporary character string pointer
2     Unit_number,                  ! Storage for unit number
2     Unit_len ;
2
2 !
2 ! Set up the necessary pointers
2 !
2 Bind
2     Ptr_a = CH$PTR (UPLIT ('A')),
2     Ptr_z = CH$PTR (UPLIT ('Z')),
```

```
;1366    1800  2      Ptr_0 = CH$PTR (UPLIT ('0')),
;1367    1801  2      Ptr_9 = CH$PTR (UPLIT ('9')),
;1368    1802  2      Ptr_star = CH$PTR (UPLIT ('*')),
;1369    1803  2      Ptr_colon = CH$PTR (UPLIT (':')) ;
;1370    1804  2
;1371    1805  2
;1372    1806  2  !
;1373    1807  2  ! Determine if the string specified falls in the range of
;1374    1808  2  ! the minimum/maximum possible length.
;1375    1809  2  !
;1376    1810  2  If (.name[dsc$w_length] LSS min_len) OR
;1377    1811  3      (.name[dsc$w_length] GTR max_len)
;1378    1812  2  Then
;1379    1813  2      !
;1380    1814  2      ! The string is either too small or too large, indicate that
;1381    1815  2      ! this is invalid input by returning to the calling routine
;1382    1816  2      ! with a false value.
;1383    1817  2      !
;1384    1818  2      Return false ;
;1385    1819  2
;1386    1820  2  !
;1387    1821  2  ! Parse the device name specification.
;1388    1822  2  !
;1389    1823  2  ! Set up pointer to the end of string and break out the
;1390    1824  2  ! unit number designation.
;1391    1825  2  !
;1392    1826  2  Ptr = CH$PTR (.name[dsc$a_pointer],.name[dsc$w_length]-1) ;
;1393    1827  2  Str_len = 0 ;
;1394    1828  2  Unit_len = 0 ;
;1395    1829  2  Sp_chr_len = 0 ;
;1396    1830  2
;1397    1831  2  Until (CH$GEQ (1,.ptr,1,ptr_a)) Do
;1398    1832  3      Begin
;1399    1833  3      If (CH$GEQ (1,.ptr,1,ptr_0)) AND
;1400    1834  4          (CH$LEQ (1,.ptr,1,ptr_9))
;1401    1835  3      Then
;1402    1836  3          !
;1403    1837  3          ! Valid unit number, update string length and point to the
;1404    1838  3          ! next character back.
;1405    1839  3          !
;1406    1840  4          Begin
;1407    1841  4          Unit_len = .unit_len + 1 ;
;1408    1842  4          Ptr = CH$PLUS (.ptr,-1) ;
;1409    1843  4          End
;1410    1844  3      Else
;1411    1845  3          !
;1412    1846  3          ! Not a valid unit number, determine if it was a wildcard character.
;1413    1847  3          !
;1414    1848  4          Begin
;1415    1849  5          If (CH$EQL (1,.ptr,1,ptr_star))
;1416    1850  4          Then
;1417    1851  4              !
;1418    1852  4              ! Indicate  wildcard specified.
;1419    1853  4              !
;1420    18 4  5              Begin
;1421    18 5  5              Sp_chr_len = .sp_chr_len + 1 ;
;1422    1856  5              Ptr = CH$PLUS (.ptr,-1) ;
```

```
;  1423     1857  5            End
;  1424     1858  4        Else
;  1425     1859  5            Begin
;  1426     1860  6            If (CH$EQL (1,.ptr,1,CH$PTR (uplit('$'))))
;  1427     1861  5            Then
;  1428     1862  6                Begin
;  1429     1863  6                Exitloop ;
;  1430     1864  6                End
;  1431     1865  5            Else
;  1432     1866  5                !
;  1433     1867  5                ! Not a valid unit number or wildcard designator, only other
;  1434     1868  5                ! valid input would be a colon. Determine if it was a colon.
;  1435     1869  5                !
;  1436     1870  6                Begin
;  1437     1871  7                If NOT (CH$EQL (1,.ptr,1,ptr_colon))
;  1438     1872  6                Then
;  1439     1873  6                    !
;  1440     1874  6                    ! Indicate that this is invalid input, by returning to the
;  1441     1875  6                    ! calling routine with a false value.
;  1442     1876  6                    !
;  1443     1877  6                    Return false
;  1444     1878  6                Else
;  1445     1879  6                    !
;  1446     1880  6                    ! Character was a colon, point to the next character back.
;  1447     1881  6                    !
;  1448     1882  7                    Begin
;  1449     1883  7                    Sp_chr_len = .sp_chr_len + 1 ;
;  1450     1884  7                    Ptr = CH$PLUS (.ptr,-1) ;
;  1451     1885  6                    End ;
;  1452     1886  5                End ;
;  1453     1887  4            End ;
;  1454     1888  3        End ;
;  1455     1889  2    End ;
;  1456     1890  2
;  1457     1891  2 !
;  1458     1892  2 ! Ensure that the unit number designation is not larger then 5 digits,
;  1459     1893  2 ! convert and save it.
;  1460     1894  2 !
;  1461     1895  2 If .unit_len NEQ 0
;  1462     1896  2 Then
;  1463     1897  3     Begin
;  1464     1898  4     If (.unit_len GTRU unit_number_len)
;  1465     1899  3     Then
;  1466     1900  3         !
;  1467     1901  3         ! Unit number designation too large. Return to calling routine.
;  1468     1902  3         !
;  1469     1903  3         Return false
;  1470     1904  3     Else
;  1471     1905  3         !
;  1472     1906  3         ! Point to the begining of the unit number(s) and convert the
;  1473     1907  3         ! ascii unit number to a binary value and save it.
;  1474     1908  3         !
;  1475     1909  4         Begin
;  1476     1910  4         Ptr_unit = CH$PLUS (.ptr,1) ;
;  1477     1911  5         If (status = LIB$CVT_DTB (.unit_len,.ptr_unit,unit_number))
;  1478     1912  4         Then
;  1479     1913  4             !
```

```
;  1480     1914  4                ! Save the unit number in the queue entry.
;  1481     1915  4                !
;  1482     1916  5                Begin
;  1483     1917  5                Dev_select[dev$w_unit] = .unit_number
;  1484     1918  5                End
;  1485     1919  4            Else
;  1486     1920  4                !
;  1487     1921  4                ! Error converting the unit number, notify user.
;  1488     1922  4                !
;  1489     1923  4                Signal (erf_cvterr, 2,.unit_len,ptr_unit) ;
;  1490     1924  3            End ;
;  1491     1925  2        End ;
;  1492     1926  2
;  1493     1927  2 !
;  1494     1928  2 ! Set the pointer to the begining of the string. Calculate remaining
;  1495     1929  2 ! string length.
;  1496     1930  2 !
;  1497     1931  2 Ptr = CH$PTR (.name[dsc$a_pointer]) ;
;  1498     1932  2 Str_len = ((.name[dsc$w_length] - .unit_len) - .sp_chr_len);
;  1499     1933  2
;  1500     1934  2 !
;  1501     1935  2 ! Determine if a node name was specified.
;  1502     1936  2 !
;  1503     1937  2 Tmp_ptr = CH$FIND_CH (.str_len,.ptr,%C'$') ;
;  1504     1938  2 If .tmp_ptr NEQ 0
;  1505     1939  2 Then
;  1506     1940  2     !
;  1507     1941  2     ! Not a null pointer, there is a node name. Update the string
;  1508     1942  2     ! pointer so it points to the device name and adjust the
;  1509     1943  2     ! string length.
;  1510     1944  2     !
;  1511     1945  3     Begin
;  1512     1946  3     Name_size = CH$DIFF (.tmp_ptr, .ptr) ;
;  1513     1947  3     If .name_size GTR 6
;  1514     1948  3     Then
;  1515     1949  3         !
;  1516     1950  3         Signal_stop (msg$_invquaval, 2,.name,include_desc) ;
;  1517     1951  3
;  1518     1952  3     Ptr = CH$PLUS (.tmp_ptr,1) ;
;  1519     1953  3     Str_len = .str_len - (.name_size + 1) ;
;  1520     1954  3     End
;  1521     1955  2 Else
;  1522     1956  2     !
;  1523     1957  2     ! Did not locate a '$' in the string, ensure the string
;  1524     1958  2     ! meets length restrictions for device name only.
;  1525     1959  2     !
;  1526     1960  3     Begin
;  1527     1961  3     !
;  1528     1962  3     ! Indicate that a node name was not specified.
;  1529     1963  3     !
;  1530     1964  3     Dev_select[dev$v_node_name_wild] = true ;
;  1531     1965  3
;  1532     1966  3     If .str_len GEQ 7
;  1533     1967  3     Then
;  1534     1968  3         !
;  1535     1969  3         ! Illegal string length, for specifying device name
;  1536     1970  3         ! without a node name.
```

```
;  1537          1971  3          !
;  1538          1972  3          Signal_stop (msg$_invquaval, 2,.name,include_desc) ;
;  1539          1973  3
;  1540          1974  2      End ;
;  1541          1975  2
;  1542          1976  2  !
;  1543          1977  2  ! Ensure tnat the device and controller designation are
;  1544          1978  2  ! valid. Save the starting pointer.
;  1545          1979  2  !
;  1546          1980  2  Tmp_ptr = .ptr ;
;  1547          1981  2
;  1548          1982  2  Incr I from 1 to .str_len do
;  1549          1983  3      Begin
;  1550          1984  3      If (CH$GEQ (1,.ptr,1,ptr_a)) AND
;  1551          1985  4          (CH$LEQ (1,.ptr,1,ptr_z))
;  1552          1986  3      Then
;  1553          1987  3          !
;  1554          1988  3          ! Valid character for this string, point to the next character.
;  1555          1989  3          !
;  1556          1990  4          Begin
;  1557          1991  4          Name_len = .name_len + 1 ;
;  1558          1992  4          Ptr = CH$PLUS (.ptr,1) ;
;  1559          1993  4          End
;  1560          1994  3       Else
;  1561          1995  3          Return false ;
;  1562          1996  3
;  1563          1997  2      End ;
;  1564          1998  2
;  1565          1999  2  !
;  1566          2000  2  ! Save the device and controller designation as
;  1567          2001  2  ! a counted ascii string.
;  1568          2002  2  !
;  1569          2003  2
;  1570          2004  2  Dev_select[dev$b_dev_name_length] =
;  1571          2005  2                  ((.name[dsc$w_length] - .unit_len) - .sp_chr_len) ;
;  1572          2006  2
;  1573          2007  2  CH$MOVE (.dev_select[dev$b_dev_name_length],
;  1574          2008  2                  .name[dsc$a_pointer],dev_select[dev$t_dev_name]) ;
;  1575          2009  2
;  1576          2010  2  !
;  1577          2011  2  ! Save the two-character device name designation seperately, for use
;  1578          2012  2  ! in correlating the device name with a device class.
;  1579          2013  2  !
;  1580          2014  2  If .str_len EQL 0
;  1581          2015  2  Then
;  1582          2016  2      ! Indicate that the device name has been wild carded.
;  1583          2017  2      !
;  1584          2018  2      Wild_carded_device = true
;  1585          2019  2  Else
;  1586          2020  2      ! There is a device name get the first two characters
;  1587          2021  2      ! so that a device class can be translated.
;  1588          2022  2      !
;  1589          2023  2      CH$MOVE (2,.tmp_ptr,dev_name) ;
;  1590          2024  2
;  1591          2025  2  If .exclude_flag
;  1592          2026  2  Then
;  1593          2027  2      !
```

```
2     ! Indicate that this device is to be excluded.
2     !
3     Begin
3     Dev_select[dev$v_exclude_flg] = true ;
3     Exclude_mask[exc$v_device_select] = true ;
3     End
2 Else
2     Include_mask[inc$v_device_select] = true ;
2
2 Return true ;
2
1 End ;          ! Routine


                                                   .PSECT  $PLIT,NOWRT,NOEXE,  PIC,2

                  00  00  00  41  00250 P.ACL:  .ASCII  \A\<0><0><0>                 ;
                  00  00  00  5A  00254 P.ACM:  .ASCII  \Z\<0><0><0>                 ;
                  00  00  00  30  00258 P.ACN:  .ASCII  \0\<0><0><0>                 ;
                  00  00  00  39  0025C P.ACO:  .ASCII  \9\<0><0><0>                 ;
                  00  00  00  2A  00260 P.ACP:  .ASCII  \*\<0><0><0>                 ;
                  00  00  00  3A  00264 P.ACQ:  .ASCII  \:\<0><0><0>                 ;
                  00  00  00  24  00268 P.ACR:  .ASCII  \$\<0><0><0>                 ;

                                        PTR_A=              P.ACL
                                        PTR_Z=              P.ACM
                                        PTR_0=              P.ACN
                                        PTR_9=              P.ACO
                                        PTR_STAR=           P.ACP
                                        PTR_COLON=          P.ACQ


                                                   .PSECT  $CODE,NOWRT,  PIC,2

                          0FFC 00000               .ENTRY  PARSE_DEVNAME, Save R2,R3,R4,R5,R6,R7,R8,-  ; 1743
                                                           R9,R10,R11                          ;
           5B 00000000'   00  9E 00002             MOVAB   PTR_A, R11                          ;
           5A 00000000'   00  9E 00009             MOVAB   DEV_SELECT, R10                     ;
           5E             08  C2 00010             SUBL2   #8, SP                              ;
                          59  D4 00013             CLRL    NAME_LEN                            ; 1767
                          54  D4 00015             CLRL    SP_CHR_LEN                          ;
           53       04    AC  D0 00017             MOVL    NAME, R3                            ; 1810
           02             63  B1 0001B             CMPW    (R3), #2                            ;
                          03  1E 0001E             BGEQU   2$                                  ;
                        0141  31 00020 1$:         BRW     20$                                 ;
           0E             63  B1 00023 2$:         CMPW    (R3), #14                           ; 1811
                          F8  1A 00026             BGTRU   1$                                  ;
           52             63  3C 00028             MOVZWL  (R3), R2                            ; 1826
           52       04    A3  C0 0002B             ADDL2   4(R3), R2                           ;
                          52  D7 0002F             DECL    PTR                                 ;
                          55  7C 00031             CLRQ    UNIT_LEN                            ; 1828
                          54  D4 00033             CLRL    SP_CHR_LEN                          ; 1829
           6B             62  91 00035 3$:         CMPB    (PTR), PTR_A                        ; 1831
                          28  1E 00038             BGEQU   7$                                  ;
      08   AB             62  91 0003A             CMPB    (PTR), PTR_0                        ; 1833
                          0A  1F 0003E             BLSSU   4$                                  ;
```

```
                        0C   AB          62  91 00040          CMPB    (PTR), PTR_9                    ; 1834
                                         04  1A 00044          BGTRU   4$                              ;
                                         55  D6 00046          INCL    UNIT_LEN                        ; 1841
                                         14  11 00048          BRB     6$                              ; 1842
                        10   AB          62  91 0004A 4$:      CMPB    (PTR), PTR_STAR                 ; 1849
                                         0C  13 0004E          BEQL    5$                              ;
                        18   AB          62  91 00050          CMPB    (PTR), P.ACR                    ; 1860
                                         0C  13 00054          BEQL    7$                              ;
                        14   AB          62  91 00056          CMPB    (PTR), PTR_COLON                ; 1871
                                         C4  12 0005A          BNEQ    1$                              ;
                                         54  D6 0005C 5$:      INCL    SP_CHR_LEN                      ; 1883
                                         52  D7 0005E 6$:      DECL    PTR                             ; 1884
                                         D3  11 00060          BRB     3$                              ; 1871
                                         55  D5 00062 7$:      TSTL    UNIT_LEN                        ; 1895
                                         38  13 00064          BEQL    9$                              ;
                            05           55  D1 00066          CMPL    UNIT_LEN, #5                    ; 1898
                                         B5  1A 00069          BGTRU   1$                              ;
                        04   AE      01  A2  9E 0006B          MOVAB   1(R2), PTR_UNIT                 ; 1910
                                         5E  DD 00070          PUSHL   SP                              ; 1911
                                     08  AE  DD 00072          PUSHL   PTR_UNIT                        ;
                                         55  DD 00075          PUSHL   UNIT_LEN                        ;
                00000000G   00           03  FB 00077          CALLS   #3, LIB$CVT_DTB                 ;
                            09           50  E9 0007E          BLBC    STATUS, 8$                      ;
                            50           6A  D0 00081          MOVL    DEV_SELECT, R0                  ; 1917
                        1B   A0          6E  B0 00084          MOVW    UNIT_NUMBER, 27(R0)             ;
                                         14  11 00088          BRB     9$                              ; 1916
                                     04  AE  9F 0008A 8$:      PUSHAB  PTR_UNIT                        ; 1923
                                         55  DD 0008D          PUSHL   UNIT_LEN                        ;
                                         02  DD 0008F          PUSHL   #2                              ;
                            00000000G    8F  DD 00091          PUSHL   #ERF_CVTERR                     ;
                00000000G   00           04  FB 00097          CALLS   #4, LIB$SIGNAL                  ;
                            52       04  A3  D0 0009E 9$:      MOVL    4(R3), PTR                      ; 1931
                            50           63  3C 000A2          MOVZWL  (R3), R0                        ; 1932
                            50           55  C2 000A5          SUBL2   UNIT_LEN, R0                    ;
56                          50           54  C3 000A8          SUBL3   SP_CHR_LEN, R0, STR_LEN         ;
62                          56           24  3A 000AC          LOCC    #36, STR_LEN, (PTR)             ; 1937
                                         02  12 000B0          BNEQ    10$                             ;
                                         51  D4 000B2          CLRL    R1                              ;
                            58           51  D0 000B4 10$:     MOVL    R1, TMP_PTR                     ;
                                         2E  13 000B7          BEQL    12$                             ; 1938
57                          58           52  C3 000B9          SUBL3   PTR, TMP_PTR, NAME_SIZE         ; 1946
                            06           57  D1 000BD          CMPL    NAME_SIZE, #6                   ; 1947
                                         17  15 000C0          BLEQ    11$                             ;
                                    00000000G    00  9F 000C2          PUSHAB  INCLUDE_DESC            ; 1950
                                         53  DD 000C8          PUSHL   R3                              ;
                                         02  DD 000CA          PUSHL   #2                              ;
                                    0008132C     8F  DD 000CC          PUSHL   #529196                 ;
                00000000G   00           04  FB 000D2          CALLS   #4, LIB$STOP                    ;
                            52       01  A8  9E 000D9 11$:     MOVAB   1(R8), PTR                      ; 1952
50                          56           57  C3 000DD          SUBL3   NAME_SIZE, STR_LEN, R0          ; 1953
                            56       FF  A0  9E 000E1          MOVAB   -1(R0), STR_LEN                 ;
                                         23  11 000E5          BRB     13$                             ; 1938
                            50           6A  D0 000E7 12$:     MOVL    DEV_SELECT, R0                  ; 1964
                        1E   A0          01  88 000EA          BISB2   #1, 30(R0)                      ;
                            07           56  D1 000EE          CMPL    STR_LEN, #7                     ; 1966
                                         17  19 000F1          BLSS    13$                             ;
                                    00000000G    00  9F 000F3          PUSHAB  INCLUDE_DESC            ; 1972
```

```
                                        53  DD 000F9        PUSHL   R3                                ;
                                        02  DD 000FB        PUSHL   #2                                ;
                              0008132C  8F  DD 000FD        PUSHL   #529196                           ;
                  00000000G  00         04  FB 00103        CALLS   #4, LIB$STOP                      ;
                             58         52  D0 0010A  13$:  MOVL    PTR, TMP_PTR                      ; 1980
                                        50  D4 0010D        CLRL    I                                 ; 1984
                                        0F  11 0010F        BRB     15$                               ;
                             6B         62  91 00111  14$:  CMPB    (PTR), PTR_A                      ;
                                        4E  1F 00114        BLSSU   20$                               ;
                        04   AB         62  91 00116        CMPB    (PTR), PTR_Z                      ; 1985
                                        48  1A 0011A        BGTRU   20$                               ;
                                        59  D6 0011C        INCL    NAME_LEN                          ; 1991
                                        52  D6 0011E        INCL    PTR                               ; 1992
            ED               50         56  F3 00120  15$:  AOBLEQ  STR_LEN, I, 14$                   ; 1982
                             57         6A  D0 00124        MOVL    DEV_SELECT, R7                    ; 2004
                             52         63  3C 00127        MOVZWL  (R3), R2                          ; 2005
                             52         55  C2 0012A        SUBL2   UNIT_LEN, R2                      ;
        08  A7               52         54  83 0012D        SUBB3   SP_CHR_LEN, R2, 8(R7)             ;
                             50    08   A7  98 00132        CVTBL   8(R7), R0                         ; 2007
        09  A7          04   B3         50  28 00136        MOVC3   R0, @4(R3), 9(R7)                 ; 2008
                                        56  D5 0013C        TSTL    STR_LEN                           ; 2014
                                        06  12 0013E        BNEQ    16$                               ;
                        4C   AA         01  D0 00140        MOVL    #1, WILD_CARDED_DEVICE            ; 2018
                                        04  11 00144        BRB     17$                               ;
                        FC   AA         68  B0 00146  16$:  MOVW    (TMP_PTR), DEV_NAME               ; 2023
                             0A    0E   AA  E9 0014A  17$:  BLBC    EXCLUDE_FLAG, 18$                 ; 2025
                        1E   A7         02  88 0014E        BISB2   #2, 30(R7)                        ; 2031
                             50    18   AA  D0 00152        MOVL    EXCLUDE_MASK, R0                  ; 2032
                                        04  11 00156        BRB     19$                               ;
                             50    30   AA  D0 00158  18$:  MOVL    INCLUDE_MASK, R0                  ; 2035
                        02   A0         10  88 0015C  19$:  BISB2   #16, 2(R0)                        ;
                             50         01  D0 00160        MOVL    #1, R0                            ; 2037
                                            04 00163        RET                                       ;
                                        50  D4 00164  20$:  CLRL    R0                                ; 2039
                                            04 00166        RET                                       ;

; Routine Size:  359 bytes,    Routine Base:  $CODE + 070C

; 1606        2040  1
```

```
; 1608          2041  1 ROUTINE CLASS_OPTION_CHECK: NOVALUE =
; 1609          2042  1
; 1610          2043  1 !++
; 1611          2044  1 !
; 1612          2045  1 ! Functional Description:
; 1613          2046  1 !
; 1614          2047  1 !    This routine verifies that there are no conflicts between /exclude and
; 1615          2048  1 !    /include device name and class option selections. Following is a set of
; 1616          2049  1 !    example inputs and a brief description of how they are handled:
; 1617          2050  1 !
; 1618          2051  1 !    /include=MF,TAPE (the queue entry for MF will be removed because that
; 1619          2052  1 !                      entire class of devices is selected (tape)).
; 1620          2053  1 !    /exclude=MF,TAPE (the queue entry for MF will be removed because that
; 1621          2054  1 !                      entire class of devices is selected (tape)).
; 1622          2055  1 !
; 1623          2056  1 !    /include=TAPE
; 1624          2057  1 !     and /exclude=MF (valid command - will output all tape entries except
; 1625          2058  1 !                      MF entries)
; 1626          2059  1 !
; 1627          2060  1 !    /include=MF
; 1628          2061  1 !     and /exclude=TAPE (The /include option will take precedence over the
; 1629          2062  1 !                      /exclude option and the tape class indicator will
; 1630          2063  1 !                      be cleared.)
; 1631          2064  1 !
; 1632          2065  1 ! Calling Sequence:
; 1633          2066  1 !
; 1634          2067  1 !    CLASS_OPTION_CHECK () ;
; 1635          2068  1 !
; 1636          2069  1 ! Input Parameters:
; 1637          2070  1 !
; 1638          2071  1 !    None
; 1639          2072  1 !
; 1640          2073  1 ! Output Parameters:
; 1641          2074  1 !
; 1642          2075  1 !    None
; 1643          2076  1 !
; 1644          2077  1 !--
; 1645          2078  2 Begin
; 1646          2079  2
; 1647          2080  2 MAP
; 1648          2081  2     Dev_select:         REF $BBLOCK,
; 1649          2082  2     Exclude_class:      REF VECTOR[,byte],
; 1650          2083  2     Exclude_key:        REF VECTOR,
; 1651          2084  2     Exclude_mask:       REF $BBLOCK,
; 1652          2085  2     Include_class:      REF VECTOR[,byte],
; 1653          2086  2     Include_key:        REF VECTOR,
; 1654          2087  2     Include_mask:       REF $BBLOCK ;
; 1655          2088  2
; 1656          2089  2 LOCAL
; 1657          2090  2     Status ;
; 1658          2091  2
; 1659          2092  2 !
; 1660          2093  2 ! Both the /include and /exclude qualifiers have been parsed. Determine
; 1661          2094  2 ! if there are any conflicts between the devices and device classes
; 1662          2095  2 ! that have been included / excluded.
; 1663          2096  2 !
; 1664          2097  2 ! Ensure queue is not empty
```

```
;1665     2098  2 !
;1666     2099  2 If .que_entry_cnt LEQ 0
;1667     2100  2 Then
;1668     2101  2     !
;1669     2102  2     ! Exit, empty queue
;1670     2103  2     !
;1671     2104  2     Return ;
;1672     2105  2
;1673     2106  2 !
;1674     2107  2 ! Determine if there were both device and device class entries selected.
;1675     2108  2 !
;1676     2109  3 If ((NOT .include_mask[inc$v_dev_class_select]) AND
;1677     2110  2     (NOT .include_mask[inc$v_device_select])) OR
;1678     2111  3    ((NOT .exclude_mask[exc$v_dev_class_select]) AND
;1679     2112  3     (NOT .exclude_mask[exc$v_device_select]))
;1680     2113  2 Then
;1681     2114  2     !
;1682     2115  2     ! Either one of the other is not selected, return to calling
;1683     2116  2     ! routine.
;1684     2117  2     !
;1685     2118  2     Return ;
;1686     2119  2
;1687     2120  2 !
;1688     2121  2 ! Get the address of the first entry in the queue.
;1689     2122  2 !
;1690     2123  2 Que_entry_addrs = root_flink + .root_flink ;
;1691     2124  2
;1692     2125  2 !
;1693     2126  2 ! Read an entry from the queue.
;1694     2127  2 !
;1695     2128  2 Incr I from 1 to .que_entry_cnt do
;1696     2129  3     Begin
;1697     2130  3     !
;1698     2131  3     ! Determine if either the exclude or include device class
;1699     2132  3     ! selections conflict with any of the devices selected.
;1700     2133  3     ! (/include,/exclude=tapes and/or /include,/exclude=MF)
;1701     2134  3     !
;1702     2135  3     Incr J from 0 to max_class do
;1703     2136  4         Begin
;1704     2137  4         If .exclude_mask[exc$v_dev_class_select]
;1705     2138  4         Then
;1706     2139  4             ! Determine if exclude bit recorded in the entry was set.
;1707     2140  5             Begin
;1708     2141  6             If ( (.exclude_class[.J] EQL .que_entry_addrs[dev$b_class]) AND
;1709     2142  6                 (.que_entry_addrs[dev$v_exclude_flg]) )
;1710     2143  5             Then
;1711     2144  5                 !
;1712     2145  5                 ! Remove entry from queue because the entire class
;1713     2146  5                 ! of devices is excluded.
;1714     2147  5                 !
;1715     2148  6                 Begin
;1716     2149  7                 If NOT (status = LIB$REMQTI (root_flink,.que_entry_addrs))
;1717     2150  6                 Then
;1718     2151  6                     Signal (.status) ;
;1719     2152  6
;1720     2153  6                 !
;1721     2154  6                 ! Update the que entry count, determine if there are
```

```
;  1722     2155  6              ! any device selections left, and return to calling routine.
;  1723     2156  6              !
;  1724     2157  6              Exclude_q_entry_cnt = .exclude_q_entry_cnt - 1 ;
;  1725     2158  6
;  1726     2159  6              If .que_entry_cnt EQL 0
;  1727     2160  6              Then
;  1728     2161  6                  !
;  1729     2162  6                  ! Indicate that there are no device selections.
;  1730     2163  6                  !
;  1731     2164  7                  Begin
;  1732     2165  7                  Exclude_mask[exc$v_device_select] = false ;
;  1733     2166  6                  End ;
;  1734     2167  6
;  1735     2168  6              Return ;
;  1736     2169  6              End
;  1737     2170  5          Else
;  1738     2171  5              !
;  1739     2172  5              ! Conflicting /include and /exclude option
;  1740     2173  5              ! selections. Reset the exclude class selection.
;  1741     2174  5              ! Determine if there are any device class selections left,
;  1742     2175  5              ! notify the user and return.
;  1743     2176  5              !
;  1744     2177  6              Begin
;  1745     2178  6              Exclude_mask[0,.(exclude_key+.J),1,0] = false ;
;  1746     2179  6
;  1747     2180  6              !
;  1748     2181  6              ! Indicate that there are no device class
;  1749     2182  6              ! selections made for /exclude.
;  1750     2183  6              !
;  1751     2184  6              Exclude_mask[exc$v_dev_class_select] = false ;
;  1752     2185  6              Return ;
;  1753     2186  5              End ;
;  1754     2187  4          End ;
;  1755     2188  4
;  1756     2189  4          If .include_mask[inc$v_dev_class_select]
;  1757     2190  4          Then
;  1758     2191  4              !
;  1759     2192  4              ! Remove the entry from the queue because either (1)entire class
;  1760     2193  4              ! of devices is includes OR (2)conflicting /include and
;  1761     2194  4              ! /exclude options were selected (The /include selection has
;  1762     2195  4              ! precedence over the /exclude selection).
;  1763     2196  4              !
;  1764     2197  5              Begin
;  1765     2198  6              If ( (.include_class[.J] EQL .que_entry_addrs[dev$b_class]) AND
;  1766     2199  6                  (NOT .que_entry_addrs[dev$v_exclude_flg]) )
;  1767     2200  5              Then
;  1768     2201  5                  !
;  1769     2202  5                  !
;  1770     2203  5                  !
;  1771     2204  6                  Begin
;  1772     2205  7                  If NOT (status = LIB$REMQTI (root_flink,.que_entry_addrs))
;  1773     2206  6                  Then
;  1774     2207  6                      Signal (.status) ;
;  1775     2208  6
;  1776     2209  6                  !
;  1777     2210  6                  ! Update the que entry count, and determine if there are
;  1778     2211  6                  ! any device selections left.
```

```
;  1779          2212  6                   !
;  1780          2213  6                   Include_q_entry_cnt = .include_q_entry_cnt - 1 ;
;  1781          2214  6                   If .que_entry_cnt EQL 0
;  1782          2215  6                   Then
;  1783          2216  6                       !
;  1784          2217  6                       ! Indicate that there are no device selections.
;  1785          2218  6                       !
;  1786          2219  7                       Begin
;  1787          2220  7                       Include_mask[inc$v_device_select] = false ;
;  1788          2221  6                       End ;
;  1789          2222  6
;  1790          2223  6                   Return ;
;  1791          2224  5                   End ;
;  1792          2225  4               End ;
;  1793          2226  3           End ;
;  1794          2227  3
;  1795          2228  3       !
;  1796          2229  3       ! Update the que entry address, to get the next entry.
;  1797          2230  3       !
;  1798          2231  3       Que_entry_addrs = .que_entry_addrs + .que_entry_addrs[dev$a_flink] ;
;  1799          2232  2       End ;
;  1800          2233  1 End ;                ! Routine


                                        OFFC 00000 CLASS_OPTION_CHECK:
                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 2041
                                5B 00000000G    00  9E 00002        MOVAB   LIB$SIGNAL, R11             ;
                                5A 00000000G    00  9E 00009        MOVAB   LIB$REMQTI, R10             ;
                                59     0000'    CF  9E 0C010        MOVAB   QUE_ENTRY_ADDRS, R9         ;
                                58 00000000'    00  9E 00015        MOVAB   QUE_ENTRY_CNT, R8           ;
                                55              68  3C 0001C        MOVZWL  QUE_ENTRY_CNT, R5           ; 2099
                                                01  14 0001F        BGTR    1$                          ;
                                                04  00021        RET                                 ;
                                50      F0      A8  D0 00022 1$:    MOVL    INCLUDE_MASK, R0            ; 2109
                          04    60              15  E0 00026        BBS     #21, (R0), 2$               ;
                          08    60              14  E1 0002A        BBC     #20, (R0), 3$               ; 2110
                                50      D8      A8  D0 0002E 2$:    MOVL    EXCLUDE_MASK, R0            ; 2111
                          05    60              15  E0 00032        BBS     #21, (R0), 4$               ;
                          01    60              14  E0 00036 3$:    BBS     #20, (R0), 4$               ; 2112
                                                04  0003A        RET                                 ;
                                50      F4      A9  9E 0003B 4$:    MOVAB   ROOT_FLINK, R0              ; 2123
                                69      F4 B940     9E 0003F        MOVAB   @ROOT_FLINK[R0], QUE_ENTRY_ADDRS
                                                56  D4 00044        CLRL    I                           ; 2128
                                                009B 31 00046       BRW     14$                         ;
                                                53  D4 00049 5$:    CLRL    J                           ; 2135
                                52      D8      A8  D0 0004B 6$:    MOVL    EXCLUDE_MASK, R2            ; 2137
                          45    62              15  E1 0004F        BBC     #21, (R2), 10$              ;
                                51      C8      A8  D0 00053        MOVL    EXCLUDE_CLASS, R1           ; 2141
                                50              69  D0 00057        MOVL    QUE_ENTRY_ADDRS, R0         ;
                                57      1D      A0  98 0005A        CVTBL   29(R0), R7                  ;
                57      6341'   08              00  ED 0005E        CMPZV   #0, #8, (J)[R1], R7         ;
                                                25  12 00064        BNEQ    8$                          ;
                          20    1E      A0          01  E1 00066    BBC     #1, 30(R0), 8$              ; 2142
                                                50  DD 0006B        PUSHL   R0                          ; 2149
```

```
                              F4   A9  9F 0006D         PUSHAB  ROOT_FLINK                          ;
                    6A             02  FB 00070         CALLS   #2, [IB$REMQTI                      ;
                    54             50  D0 00073         MOVL    R0, STATUS                          ;
                    05             54  E8 00076         BLBS    STATUS, 7$                          ;
                                   54  DD 00079         PUSHL   STATUS                              ; 2151
                    6B             01  FB 0007B         CALLS   #1, LIB$SIGNAL                      ;
                              DC   A8  97 0007E  7$:    DECB    EXCLUDE_Q_ENTRY_CNT                 ; 2157
                                   68  B5 00081         TSTW    QUE_ENTRY_CNT                       ; 2159
                                   65  12 00083         BNEQ    15$                                 ;
                    50        D8   A8  D0 00085         MOVL    EXCLUDE_MASK, R0                    ; 2165
                                   4B  11 00089         BRB     12$                                 ;
                              D0 A843  9F 0008B  8$:    PUSHAB  EXCLUDE_KEY[J]                      ; 2178
          00        62             9E  E5 0008F         BBCC    @(SP)+, (R2), 9$                    ;
               02   A2             20  8A 00093  9$:    BICB2   #32, 2(R2)                          ; 2184
                                   04  00097            RET                                         ; 2177
                    50        F0   A8  D0 00098  10$:   MOVL    INCLUDE_MASK, R0                    ; 2189
          3B        60             15  E1 0009C         BBC     #21, (R0), 13$                      ;
                    51        E0   A8  D0 000A0         MOVL    INCLUDE_CLASS, R1                   ; 2198
                    50             69  D0 000A4         MOVL    QUE_ENTRY_ADDRS, R0                 ;
                    52        1D   A0  98 000A7         CVTBL   29(R0), R2                          ;
   52     6341      08             00  ED 000AB         CMPZV   #0, #8, (J)[R1], R2                 ;
                                   28  12 000B1         BNEQ    13$                                 ;
          23   1E   A0             01  E0 000B3         BBS     #1, 30(R0), 13$                     ; 2199
                                   50  DD 000B8         PUSHL   R0                                  ; 2205
                              F4   A9  9F 000BA         PUSHAB  ROOT_FLINK                          ;
                    6A             02  FB 000BD         CALLS   #2, [IB$REMQTI                      ;
                    54             50  D0 000C0         MOVL    R0, STATUS                          ;
                    05             54  E8 000C3         BLBS    STATUS, 11$                         ;
                                   54  DD 000C6         PUSHL   STATUS                              ; 2207
                    6B             01  FB 000C8         CALLS   #1, LIB$SIGNAL                      ;
                              DD   A8  97 000CB  11$:   DECB    INCLUDE_Q_ENTRY_CNT                 ; 2213
                                   68  B5 000CE         TSTW    QUE_ENTRY_CNT                       ; 2214
                                   18  12 000D0         BNEQ    15$                                 ;
                    50        F0   A8  D0 000D2         MOVL    INCLUDE_MASK, R0                    ; 2220
               02   A0             10  8A 000D6  12$:   BICB2   #16, 2(R0)                          ;
                                   04  000DA            RET                                         ; 2204
   FF6A     53      01             05  F1 000DB  13$:   ACBL    #5, #1, J, 6$                       ; 2135
                    79             99  C0 000E1         ADDL2   @QUE_ENTRY_ADDRS, QUE_ENTRY_ADDRS   ; 2231
   FF5F     56      01             55  F1 000E4  14$:   ACBL    R5, #1, I, 5$                       ; 2128
                                   04  000EA     15$:   RET                                         ; 2233

; Routine Size:  235 bytes,    Routine Base:  $CODE + 0873

; 1801          2234  1
```

```
;  1803     2235  1 GLOBAL ROUTINE DEVICE_OPTION_CHECK =
;  1804     2236  1
;  1805     2237  1 !++
;  1806     2238  1 !
;  1807     2239  1 ! Functional Description:
;  1808     2240  1 !
;  1809     2241  1 !    This routine verifies that there are no conflicts between;
;  1810     2242  1 !    (1) /exclude and /include device name selections, (2) any of the
;  1811     2243  1 !    'values' specified for either /include or /exclude. Following is a
;  1812     2244  1 !    set of example inputs and a brief description of how they are handled:
;  1813     2245  1 !
;  1814     2246  1 !    /include=MF,MF (only one entry for MF will be in the device selection queue
;  1815     2247  1 !    /exclude=MF,MF ('')
;  1816     2248  1 !
;  1817     2249  1 !    /include=MF
;  1818     2250  1 !     and /exclude=MF (user will get an error message, invalid option selection)
;  1819     2251  1 !
;  1820     2252  1 ! Return false if like entry already exists in queue.
;  1821     2253  1 ! Return true if ok to put entry in queue.
;  1822     2254  1 ! Exits with and error message if conflicting entry already in queue.
;  1823     2255  1 !
;  1824     2256  1 ! Calling Sequence:
;  1825     2257  1 !
;  1826     2258  1 !    DEVICE_OPTION_CHECK () ;
;  1827     2259  1 !
;  1828     2260  1 ! Input Parameters:
;  1829     2261  1 !
;  1830     2262  1 !    None
;  1831     2263  1 !
;  1832     2264  1 ! Output Parameters:
;  1833     2265  1 !
;  1834     2266  1 !    None
;  1835     2267  1 !
;  1836     2268  1 ! Implicit Inputs:
;  1837     2269  1 !
;  1838     2270  1 !
;  1839     2271  1 ! Implicit Outputs:
;  1840     2272  1 !
;  1841     2273  1 !
;  1842     2274  1 !--
;  1843     2275  2 Begin
;  1844     2276  2
;  1845     2277  2 EXTERNAL
;  1846     2278  2     Exclude_desc,
;  1847     2279  2     Include_desc ;
;  1848     2280  2
;  1849     2281  2 MAP
;  1850     2282  2     Dev_select:          REF $BBLOCK ;
;  1851     2283  2
;  1852     2284  2 If NOT SEARCH_QUEUE (dev_select[dev$t_dev_name],
;  1853     2285  2     dev_select[dev$b_dev_name_length],dev_select[dev$w_unit])
;  1854     2286  2 Then
;  1855     2287  2     !
;  1856     2288  2     ! The selected entry does not match any entries already
;  1857     2289  2     ! in the queue.
;  1858     2290  2     !
;  1859     2291  2     Return true ;
```

```
; 1860          2292  2
; 1861          2293  2 !
; 1862          2294  2 ! Indicate that conflicting /exclude and /include option
; 1863          2295  2 ! selections.
; 1864          2296  2 !
; 1865          2297  2 Signal (erf_cnfquaval, 2,exclude_desc,include_desc) ;
; 1866          2298  2
; 1867          2299  2 !
; 1868          2300  2 ! Due to the error message severity (fatal) this should never
; 1869          2301  2 ! be executed but satisfies the return value business for the
; 1870          2302  2 ! compiler.
; 1871          2303  2 !
; 1872          2304  2 Return false ;
; 1873          2305  2
; 1874          2306  1 End ;          ! Routine



                                         0000 00000         .ENTRY   DEVICE_OPTION_CHECK, Save nothing       ; 2235
                     50 00000000'   00   D0 00002         MOVL     DEV_SELECT, R0                          ; 2285
                               18   A0   9F 00009         PUSHAB   27(R0)                                  ;
                               08   A0   9F 0000C         PUSHAB   8(R0)                                   ;
                               09   A0   9F 0000F         PUSHAB   9(R0)                                   ; 2284
             00000000V  00              03   FB 00012         CALLS    #3, SEARCH_QUEUE                        ; 2285
                        04              50   E8 00019         BLBS     R0, 1$                                  ;
                        50              01   D0 0001C         MOVL     #1, R0                                  ; 2291
                                         04 0001F         RET                                              ;
                     00000000G  00   9F 00020 1$:     PUSHAB   INCLUDE_DESC                            ; 2297
                     00000000G  00   9F 00026         PUSHAB   EXCLUDE_DESC                            ;
                                02   DD 0002C         PUSHL    #2                                      ;
                     00000000G  8F   DD 0002E         PUSHL    #ERF_CNFQUAVAL                          ;
             00000000G  00              04   FB 00034         CALLS    #4, LIB$SIGNAL                          ;
                                50   D4 0003B         CLRL     R0                                      ; 2304
                                         04 0003D         RET                                              ; 2306

; Routine Size:  62 bytes,    Routine Base:  $CODE + 095E


; 1875          2307  1
```

```
;1877     2308  1 GLOBAL ROUTINE SEARCH_QUEUE (name,name_length,unit_number) =
;1878     2309  1
;1879     2310  1 !++
;1880     2311  1 !
;1881     2312  1 ! Functional Description:
;1882     2313  1 !
;1883     2314  1 !    This routine will search the device name queue and determine whether
;1884     2315  1 !    the device name/unit passed to it matches any of the entries in the
;1885     2316  1 !    queue. It will return true if match on either device name/unit or
;1886     2317  1 !    return false if no match.
;1887     2318  1 !
;1888     2319  1 ! Callirg Sequence:
;1889     2320  1 !
;1890     2321  1 !    SEARCH_QUEUE (device name,device name length,unit number)
;1891     2322  1 !
;1892     2323  1 ! Input Parameters:
;1893     2324  1 !
;1894     2325  1 !    Address of device name
;1895     2326  1 !    Address of device name length
;1896     2327  1 !    Unit number
;1897     2328  1 !
;1898     2329  1 ! Output Parameters:
;1899     2330  1 !
;1900     2331  1 !    None
;1901     2332  1 !
;1902     2333  1 !--
;1903     2334  2 Begin
;1904     2335  2
;1905     2336  2 LOCAL
;1906     2337  2     Device_selected:    BYTE
;1907     2338  2                         Initial (false),
;1908     2339  2     Entry_name,
;1909     2340  2     Entry_name_size,
;1910     2341  2     I:                  WORD
;1911     2342  2                         Initial (.que_entry_cnt),
;1912     2343  2     Name_ptr,
;1913     2344  2     Ptr,
;1914     2345  2     Size ,
;1915     2346  2     Size_adj ;
;1916     2347  2
;1917     2348  2 Bind select_name_size = .name_length : byte ;
;1918     2349  2 Bind unit = .unit_number : word ;
;1919     2350  2
;1920     2351  2 !
;1921     2352  2 ! Ensure queue is not empty
;1922     2353  2 !
;1923     2354  2 If .que_entry_cnt LEQ 0
;1924     2355  2 Then
;1925     2356  2     !
;1926     2357  2     ! Exit, empty queue
;1927     2358  2     !
;1928     2359  2     Return false ;
;1929     2360  2
;1930     2361  2 !
;1931     2362  2 ! Get the address of the first entry in the queue.
;1932     2363  2 !
;1933     2364  2 Que_entry_addrs = root_flink + .root_flink ;
```

```
;  1934          2365  2
;  1935          2366  2 !
;  1936          2367  2 ! Read an entry from the queue.
;  1937          2368  2 !
;  1938          2369  2 Until (.I EQL 0) OR (.device_selected) do
;  1939          2370  3     Begin
;  1940          2371  3     !
;  1941          2372  3     ! Determine if the selected device name/controller match the
;  1942          2373  3     ! device name/controller recorded by this queue entry.
;  1943          2374  3     !
;  1944          2375  3
;  1945          2376  3     ! Determine if the node name in the queue entry was wild carded.
;  1946          2377  3     !
;  1947          2378  3     If .que_entry_addrs[dev$v_node_name_wild]
;  1948          2379  3     Then
;  1949          2380  3         !
;  1950          2381  3         ! The node name is wild carded, locate the '$' in the string
;  1951          2382  3         ! and adjust the length and starting address of the string so
;  1952          2383  3         ! that the compare will be against the device name only.
;  1953          2384  3         !
;  1954          2385  4         Begin
;  1955          2386  4         Ptr = CH$FIND_CH (..name_length,.name,%C'$') ;
;  1956          2387  4         If .ptr NEQ 0
;  1957          2388  4         Then
;  1958          2389  4             !
;  1959          2390  4             ! Found a $ in the string, compensate for node name being
;  1960          2391  4             ! logged and being wild carded when device(s) selected for
;  1961          2392  4             ! output.
;  1962          2393  4             !
;  1963          2394  5             Begin
;  1964          2395  5             Size_adj = (.ptr - .name) ;
;  1965          2396  5             .Name_length = (..name_length - .size_adj) ;
;  1966          2397  5             Name = CH$PLUS (.ptr,1) ;
;  1967          2398  4             End ;
;  1968          2399  3         End ;
;  1969          2400  3
;  1970          2401  3     Size = MINU (.que_entry_addrs[dev$b_dev_name_length],.select_name_size) ;
;  1971          2402  3
;  1972          2403  3     !
;  1973          2404  3     ! Determine if the device name/controller match.
;  1974          2405  3     !
;  1975          2406  3     If CH$EQL (.size,.name,.size,que_entry_addrs[dev$t_dev_name])
;  1976          2407  3     Then
;  1977          2408  3         !
;  1978          2409  3         ! Determine if a unit number was specified.
;  1979          2410  3         !
;  1980          2411  4         Begin
;  1981          2412  4         Device_selected = true ;
;  1982          2413  4
;  1983          2414  5         If .que_entry_addrs[dev$w_unit] NEQ (-1)
;  1984          2415  4         Then
;  1985          2416  4             !
;  1986          2417  4             ! Unit number was specified, determine if it matches.
;  1987          2418  4             !
;  1988          2419  5             Begin
;  1989          2420  5             If .unit NEQU .que_entry_addrs[dev$w_unit]
;  1990          2421  5             Then
```

```
;  1991          2422  5                    !
;  1992          2423  5                    ! Indicate that the unit number did not
;  1993          2424  5                    ! match.
;  1994          2425  5                    !
;  1995          2426  5                    Device_selected = false ;
;  1996          2427  4                End ;
;  1997          2428  4
;  1998          2429  4            If .device_selected
;  1999          2430  4            Then
;  2000          2431  4                Exitloop ;
;  2001          2432  4
;  2002          2433  3            End ;
;  2003          2434  3
;  2004          2435  3        !
;  2005          2436  3        ! Update the que entry address and decrement the number of
;  2006          2437  3        ! queue entries that have been searched.
;  2007          2438  3        !
;  2008          2439  3        Que_entry_addrs = .que_entry_addrs + .que_entry_addrs[dev$a_flink] ;
;  2009          2440  3        I = .I - 1 ;
;  2010          2441  2        End ;
;  2011          2442  2
;  2012          2443  2 !
;  2013          2444  2 ! Ensure that the device name/controller designation and unit
;  2014          2445  2 ! numbers match, determine whether the entry is for a /include
;  2015          2446  2 ! and /exclude option.
;  2016          2447  2 !
;  2017          2448  2 If .device_selected
;  2018          2449  2 Then
;  2019          2450  3     Begin
;  2020          2451  3     If .exclude_flag AND .que_entry_addrs[dev$v_exclude_flg]
;  2021          2452  3     Then
;  2022          2453  3         !
;  2023          2454  3         ! Indicate that a '/excluded' entry was found by
;  2024          2455  3         ! returning with a true value.
;  2025          2456  3         !
;  2026          2457  3         Return true
;  2027          2458  3     Else
;  2028          2459  4         Begin
;  2029          2460  5         If (NOT .exclude_flag) AND (NOT .que_entry_addrs[dev$v_exclude_flg])
;  2030          2461  4         Then
;  2031          2462  4             !
;  2032          2463  4             ! Indicate that a '/included' entry was found by
;  2033          2464  4             ! returning with a true value.
;  2034          2465  4             !
;  2035          2466  4             Return true ;
;  2036          2467  3         End ;
;  2037          2468  2     End ;
;  2038          2469  2 !
;  2039          2470  2 ! No matching entries, return to calling routine.
;  2040          2471  2 !
;  2041          2472  2 Return false ;
;  2042          2473  2
;  2043          2474  1 End ;           ! Routine
```

```
                                          07FC 00000        .ENTRY  SEARCH_QUEUE, Save R2,R3,R4,R5,R6,R7,R8,R9,-;  2308
                                                                    R10                                            :
                    5A       0000'  CF   9E 00002           MOVAB   QUE_ENTRY_ADDRS, R10                           :
                                    55   94 00007           CLRB    DEVICE_SELECTED                                ;  2334
                    50   00000000'  00   3C 00009           MOVZWL  QUE_ENTRY_CNT, R0                              ;  2342
                    58              50   B0 00010           MOVW    R0, I                                          :
                                    50   D5 00013           TSTL    R0                                             ;  2354
                                    03   14 00015           BGTR    1$                                             :
                                  0096   31 00017           BRW     12$                                            :
                    50         F4   AA   9E 0001A 1$:       MOVAB   ROOT_FLINK, R0                                 ;  2364
                    6A         F4 BA40   9E 0001E           MOVAB   @ROOT_FLINK[R0], QUE_ENTRY_ADDRS               :
                                    58   B5 00023 2$:       TSTW    I                                              ;  2369
                                    65   13 00025           BEQL    8$                                             :
                    65              55   E8 00027           BLBS    DEVICE_SELECTED, 9$                            :
                    54              6A   D0 0002A           MOVL    QUE_ENTRY_ADDRS, R4                            ;  2378
                    1D         1E   A4   E9 0002D           BLBC    30(R4), 4$                                     :
        04    BC     08   BC        24   3A 00031           LOCC    #36, @NAME_LENGTH, @NAME                       ;  2386
                                    02   12 00037           BNEQ    3$                                             :
                                    51   D4 00039           CLRL    R1                                             :
                    57              51   D0 0003B 3$:       MOVL    R1, PTR                                        :
                                    0E   13 0003E           BEQL    4$                                             ;  2387
              59    57         04   AC   C3 00040           SUBL3   NAME, PTR, SIZE_ADJ                            ;  2395
                    08   BC         59   C2 00045           SUBL2   SIZE_ADJ, @NAME_LENGTH                         ;  2396
                    04   AC    01   A7   9E 00049           MOVAB   1(R7), NAME                                    ;  2397
                    50         08   A4   98 0004E 4$:       CVTBL   8(R4), R0                                      ;  2401
   50     08    BC       08         00   ED 00052           CMPZV   #0, #8, @NAME_LENGTH, R0                       :
                                    04   1E 00058           BGEQU   5$                                             :
                    50         08   BC   9A 0005A           MOVZBL  @NAME_LENGTH, R0                               :
                    56              50   D0 0005E 5$:       MOVL    R0, SIZE                                       :
        09    A4     04   BC        56   29 00061           CMPC3   SIZE, @NAME, 9(R4)                             ;  2406
                                    1C   12 00067           BNEQ    7$                                             :
                    55              01   90 00069           MOVB    #1, DEVICE_SELECTED                            ;  2412
              FFFF  8F         1B   A4   B1 0006C           CMPW    27(R4), #-1                                    ;  2414
                                    0E   13 00072           BEQL    6$                                             :
                    50         1B   A4   32 00074           CVTWL   27(R4), R0                                     ;  2420
   50     0C    BC       10         00   ED 00078           CMPZV   #0, #16, @UNIT_NUMBER, R0                      :
                                    02   13 0007E           BEQL    6$                                             :
                                    55   94 00080           CLRB    DEVICE_SELECTED                                ;  2426
                    0A              55   E8 00082 6$:       BLBS    DEVICE_SELECTED, 9$                            ;  2429
                    6A              64   C0 00085 7$:       ADDL2   (R4), QUE_ENTRY_ADDRS                          ;  2439
                                    58   B7 00088           DECW    I                                              ;  2440
                                    97   11 0008A           BRB     2$                                             ;  2369
                    21              55   E9 0008C 8$:       BLBC    DEVICE_SELECTED, 12$                           ;  2448
                    51   00000000'  00   9A 0008F 9$:       MOVZBL  EXCLUDE_FLAG, R1                               ;  2451
                    0B              51   E9 00096           BLBC    R1, 10$                                        :
                    50              6A   D0 00099           MOVL    QUE_ENTRY_ADDRS, R0                            :
              0B    1E         A0   01   E0 0009C           BBS     #1, 30(R0), 11$                                :
                    0C              51   E8 000A1           BLBS    R1, 12$                                        ;  2460
                    50              6A   D0 000A4 10$:      MOVL    QUE_ENTRY_ADDRS, R0                            :
              04    1E         A0   01   E0 000A7           BBS     #1, 30(R0), 12$                                :
                    50              01   D0 000AC 11$:      MOVL    #1, R0                                         ;  2466
                                         04 000AF           RET                                                    :
                                    50   D4 000B0 12$:      CLRL    R0                                             ;  2474
                                         04 000B2           RET                                                    :

; Routine Size:  179 bytes,    Routine Base:  $CODE + 099C
```

; 2044 2475 1

```
;2046     2476  1 GLOBAL ROUTINE TRANSLATE_DEVICE (search_name,dev_class) =
;2047     2477  2 Begin
;2048     2478  2
;2049     2479  2 !++
;2050     2480  2 !
;2051     2481  2 ! Functional Description:
;2052     2482  2 !
;2053     2483  2 !    This routine searches the device tables to translate the
;2054     2484  2 !    known device name to a device class.
;2055     2485  2 !
;2056     2486  2 ! Calling Sequence:
;2057     2487  2 !
;2058     2488  2 !    TRANSLATE_DEVICE (search_name,dev_class)
;2059     2489  2 !
;2060     2490  2 ! Input Parameters:
;2061     2491  2 !
;2062     2492  2 !    Search name = First two characters of device name
;2063     2493  2 !
;2064     2494  2 ! Output Parameters:
;2065     2495  2 !
;2066     2496  2 !    Dev_class = Device class if device name found ELSE
;2067     2497  2 !                -1 if device name not located in table.
;2068     2498  2 !
;2069     2499  2 !    Returns true if a match occurred.
;2070     2500  2 !    Returns false if unsupported device. (This should eventually be
;2071     2501  2 !       caught and handled by the parse_devname routine.)
;2072     2502  2 !
;2073     2503  2 !--
;2074     2504  2
;2075     2505  2 EXTERNAL
;2076     2506  2     Dev_addrs_ptr:      REF VECTOR [,long],
;2077     2507  2     Dev_class_ptr:      REF VECTOR [,word],
;2078     2508  2     Max_classes:        REF VECTOR [,byte];
;2079     2509  2
;2080     2510  2 OWN
;2081     2511  2     I:          BYTE Initial (1),              ! Device address pointer index
;2082     2512  2     Max_classes_value: BYTE;
;2083     2513  2
;2084     2514  2 LOCAL
;2085     2515  2     Dev_specific_tbl:   REF VECTOR [,word], ! Device specific table address
;2086     2516  2     K:          Initial (0) ;          ! Device specific table index
;2087     2517  2
;2088     2518  2 BIND
;2089     2519  2     Cs_name = CH$PTR (uplit('CS')) ;
;2090     2520  2
;2091     2521  2 !
;2092     2522  2 ! Class dir is the address of a table that contains supported device
;2093     2523  2 ! classes and pointers to the device class specific information tables.
;2094     2524  2 !
;2095     2525  2 ! The device class specific table contains the supported device names,
;2096     2526  2 ! image name pointers (image that needs to get activated), and transfer
;2097     2527  2 ! address pointers.
;2098     2528  2 !
;2099     2529  2 ! This routine searches all of the device class specific tables until a
;2100     2530  2 ! matching device name is located, and returns the appropriate device class.
;2101     2531  2 !
;2102     2532  2 ! Loop through all of the device class specific pointers in the class dir
```

```
;  2103          2533  2 ! table.
;  2104          2534  2 !
;  2105          2535  2
;  2106          2536  2 Max_classes_value = max_classes[0];
;  2107          2537  2
;  2108          2538  2 Incr I fro  1 to .max_classes_value do
;  2109          2539  3     Begin
;  2110          2540  3     !
;  2111          2541  3     ! Get the address of a device class specific table.
;  2112          2542  3     !
;  2113          2543  3     Dev_specific_tbl = .dev_addrs_ptr[.I] ;
;  2114          2544  3
;  2115          2545  3     !
;  2116          2546  3     ! Initialize another index for the device class specific table so don't
;  2117          2547  3     ! lose the current position. Determine if the contents of the device
;  2118          2548  3     ! name field is valid OR whether the end of the device name entries
;  2119          2549  3     ! in the table has been reached.
;  2120          2550  3     !
;  2121          2551  3     K = 1 ;
;  2122          2552  3     Until (.K EQL .dev_specific_tbl[0]) do
;  2123          2553  4         Begin
;  2124          2554  4         !
;  2125          2555  4         ! Determine if the selected device name matches any of the
;  2126          2556  4         ! device names recorded in this table.
;  2127          2557  4         !
;  2128          2558  4         If CH$EQL (?, CH$PTR(.search_name), 2, CH$PTR(dev_specific_tbl[.K]))
;  2129          2559  4         Then
;  2130          2560  4             !
;  2131          2561  4             ! The device names match. Using the class dir table index,
;  2132          2562  4             ! get the corresponding device class. (The index is divided
;  2133          2563  4             ! by 2 because device classes are words and the index is for
;  2134          2564  4             ! longwords).
;  2135          2565  4             !
;  2136          2566  5             Begin
;  2137          2567  5             .Dev_class = .dev_class_ptr[.I] ;
;  2138          2568  5             Return true ;
;  2139          2569  4             End ;
;  2140          2570  4
;  2141          2571  4         !
;  2142          2572  4         ! Update the device name pointer indices.
;  2143          2573  4         !
;  2144          2574  4         K = .K + 1 ;
;  2145          2575  3         End ;
;  2146          2576  2     End ;
;  2147          2577  2
;  2148          2578  2
;  2149          2579  2 !
;  2150          2580  2 ! The name for the console device 'CSA' is not included in the device name
;  2151          2581  2 ! tables contained in ERFLIB.TLB. It really is a second device name for
;  2152          2582  2 ! the RX device which is included in the device tables. There should be
;  2153          2583  2 ! a table that includes devices like these, however because there is only
;  2154          2584  2 ! one at this time, it is checked for explicitly.
;  2155          2585  2 !
;  2156          2586  2 If CH$EQL (2, CH$PTR(.search_name), 2, cs_name)
;  2157          2587  2 Then
;  2158          2588  2     !
;  2159          2589  2     ! Return the device class.
```

```
; 2160          2590  2         !
; 2161          2591  3         Begin
; 2162          2592  3         .Dev_class = DC$_DISK ;
; 2163          2593  3         Return true ;
; 2164          2594  2         End ;
; 2165          2595  2
; 2166          2596  2   !
; 2167          2597  2   ! Could not locate a class for this device name.
; 2168          2598  2   !
; 2169          2599  2   .Dev_class = -1 ;
; 2170          2600  2   Return false ;
; 2171          2601  2
; 2172          2602  1   End ;             ! Routine


                                                                 .PSECT  $PLIT,NOWRT,NOEXE,  PIC,2

                                  00  00  53  43  0026C P.ACS:   .ASCII  \CS\<0><0>                            ;

                                                                 .PSECT  $OWN$,NOEXE,  PIC,2

                                              01  000C0 I:       .BYTE   1                                     ;
                                                  000C1 MAX_CLASSES_VALUE:
                                                                 .BLKB   1

                                                        CS_NAME=         P.ACS
                                                                 .EXTRN  DEV_ADDRS_PTR, DEV_CLASS_PTR
                                                                 .EXTRN  MAX_CLASSES

                                                                 .PSECT  $CODE,NOWRT,  PIC,2

                                            003C  00000          .ENTRY  TRANSLATE_DEVICE, Save R2,R3,R4,R5    ; 2476
                        55 00000000'    00  9E  00002            MOVAB   MAX_CLASSES_VALUE, R5                 ;
                                        52  D4  00009            CLRL    K                                     ; 2477
                        65 00000000G    00  90  0000B            MOVB    MAX_CLASSES, MAX_CLASSES_VALUE        ; 2536
                        54              65  9A  00012            MOVZBL  MAX_CLASSES_VALUE, R4                 ; 2538
                                        50  D4  00015            CLRL    I                                     ; 2558
                                        2E  11  00017            BRB     4$                                    ;
                        51 00000000G    00  D0  00019 1$:        MOVL    DEV_ADDRS_PTR, R1                     ; 2543
                        53            6140  D0  00020            MOVL    (R1)[I], DEV_SPECIFIC_TBL             ;
                        52              01  D0  00024            MOVL    #1, K                                 ; 2551
      52          63    10              00  ED  00027 2$:        CMPZV   #0, #16, (DEV_SPECIFIC_TBL), K        ; 2552
                                        19  13  0002C            BEQL    4$                                    ;
                      6342        04    BC  B1  0002E            CMPW    @SEARCH_NAME, (DEV_SPECIFIC_TBL)[K]   ; 2558
                                        0E  12  00033            BNEQ    3$                                    ;
                        51 00000000G    00  D0  00035            MOVL    DEV_CLASS_PTR, R1                     ; 2567
                  08    BC            6140  3C  0003C            MOVZWL  (R1)[I], @DEV_CLASS                   ;
                                        16  11  00041            BRB     5$                                    ; 2568
                                        52  D6  00043 3$:        INCL    K                                     ; 2574
                                        E0  11  00045            BRB     2$                                    ; 2552
            CE          50              54  F3  00047 4$:        AOBLEQ  R4, I, 1$                             ; 2538
               00000000'  00      04    BC  B1  0004B            CMPW    @SEARCH_NAME, CS_NAME                 ; 2586
                                        08  12  00053            BNEQ    6$                                    ;
                  08    BC              01  D0  00055            MOVL    #1, @DEV_CLASS                        ; 2592
                        50              01  D0  00059 5$:        MOVL    #1, R0                                ; 2593
                                            04  0005C            RET                                           ;
```

```
                08    BC             01  CE 0005D 6$:     MNEGL   #1, @DEV_CLASS                       ; 2599
                                     50  D4 00061         CLRL    R0                                   ; 2600
                                         04 00063         RET                                          ; 2602

; Routine Size:  100 bytes,    Routine Base:  $CODE + 0A4F

; 2173          2603  1
```

```
;  2175     2604  1 GLOBAL ROUTINE GET_VM (size) =
;  2176     2605  1
;  2177     2606  1 !++
;  2178     2607  1 !
;  2179     2608  1 ! Functional Description:
;  2180     2609  1 !
;  2181     2610  1 !    This routine calls the LIB$GET_VM library routine to allocate the
;  2182     2611  1 !    requested amount of virtual memory. If the request completed
;  2183     2612  1 !    successfully the allocated area is cleared, else an error is notified.
;  2184     2613  1 !
;  2185     2614  1 ! Calling Sequence:
;  2186     2615  1 !
;  2187     2616  1 !    Base_adr = GET_VM (size)
;  2188     2617  1 !
;  2189     2618  1 ! Input Parameters:
;  2190     2619  1 !
;  2191     2620  1 !   Size in bytes) of the area to be allocated.
;  2192     2621  1 !
;  2193     2622  1 ! Output Parameters:
;  2194     2623  1 !
;  2195     2624  1 !    Base address of the allocated area (address of the first byte).
;  2196     2625  1 !
;  2197     2626  1 !--
;  2198     2627  2 Begin
;  2199     2628  2
;  2200     2629  2 LOCAL
;  2201     2630  2     Base_addrs,                    ! Storage for returned base address
;  2202     2631  2     Status ;                       ! Storage for the return status
;  2203     2632  2
;  2204     2633  2 !
;  2205     2634  2 ! Call the LIB$GET_VM routine to allocate the requested amount of
;  2206     2635  2 ! virtual memory and if it was not successful, notify the user and exit.
;  2207     2636  2 !
;  2208     2637  2 Status = LIB$GET_VM (size,base_addrs) ;
;  2209     2638  2
;  2210     2639  2 If NOT .status
;  2211     2640  2 Then
;  2212     2641  2     Signal (.status) ;
;  2213     2642  2
;  2214     2643  2 !
;  2215     2644  2 ! Clear the allocated area and return the base address of the area
;  2216     2645  2 ! to the calling routine.
;  2217     2646  2 !
;  2218     2647  2 CH$FILL (0, .size, .base_addrs) ;
;  2219     2648  2 .Base_addrs
;  2220     2649  1 End ;


                                  003C 00000          .ENTRY  GET_VM, Save R2,R3,R4,R5        ; 2604
                          5E      04   C2 00002          SUBL2   #4, SP                          ;
                                  5E   DD 00005          PUSHL   SP                              ; 2637
                              04  AC   9F 00007          PUSHAB  SIZE                            ;
                  00000000G   00      02   FB 0000A      CALLS   #2, LIB$GET_VM                  ;
                              09      50   E8 00011      BLBS    STATUS, 1$                      ; 2639
```

```
                                          50  DD 00014          PUSHL   STATUS                                    ; 2641
                        0000000G  00      01  FB 00016          CALLS   #1, LIB$SIGNAL                            ;
    04   AC           00          6E      00  2C 0001D 1$:      MOVC5   #0, (SP), #0, SIZE, @BASE_ADDRS           ; 2647
                                      00  BE     00023                                                            ;
                                  50      6E  D0 00025          MOVL    BASE_ADDRS, R0                            ; 2649
                                              04 00028          RET                                               ;

; Routine Size:  41 bytes,    Routine Base:  $CODE + 0AB3


; 2221          2650  1
; 2222          2651  1 End
; 2223          2652  0 ELUDOM



                                                          .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                        Bytes                       Attributes
;
;  QUEUE_DATA                          16  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(3)
;  $GLOBAL$                            92  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  $OWN$                              194  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  $PLIT                              624  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  $CODE                             2780  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  .  ABS  .                            0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)



;                          Library Statistics
;
;                                               -------- Symbols --------      Pages        Processing
;       File                                    Total    Loaded   Percent      Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1                18619        32         0      1000           00:01.8




;                               COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:ERFPARSER/OBJ=OBJ$:ERFPARSER MSRC$:ERFPARSER/UPDATE=(ENH$:ERFPARSER)

; Size:          2780 code + 926 data bytes
; Run Time:         00:56.6
; Elapsed Time:     01:58.5
; Lines/CPU Min:      2810
; Lexemes/CPU-Min: 19666
; Memory Used:  363 pages
; Compilation Complete
```

ERFCOM
LIS

ERFMSG
LIS

ERF
LIS

ERF5VECT
LIS

ERF2VECT
LIS

ERFOUTPUT
LIS

ERFCOMVEC
LIS

ERFBRIVEC
LIS

ERF3VECT
LIS

ERFDISKVE
LIS

ERFPARSER
LIS

ERFBUSVEC
LIS

ERFINIVEC
LIS

ERF4VECT
LIS

ERFINICOM
LIS

ERF1VECT
LIS

GETCODE.LIS

INITPROC1.LIS

INITREAL.LIS

EXECIMAGE.LIS

ERFSUMM.LIS

INITBUS.LIS

INITPROC4.LIS

KMS3271.LIS

ERFTAPEVE.LIS

HEADER.LIS

FILES.LIS

ERLLOGSYS.LIS

INITPROC2.LIS

INIT_TAPE.LIS

ERLLOGMSG.LIS

INITDISK.LIS

INITPROC5.LIS

IMAGELOAD.LIS

INITPROC3.LIS

INTERVENE.LIS

ERFRTVEC.LIS

ERFSUMVEC.LIS