```
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
EEEEEEEEEEEEEFEE RRR          RRR   FFF
EEEEEEEEEEEEEEE  RRR          RRR   FFF
EEEEEEEEEEEEEEE  RRR          RRR   FFF
```

```
RRRRRRRR    EEEEEEEEEE    CCCCCCCC    SSSSSSSS  EEEEEEEEEE  LL          EEEEEEEEEE    CCCCCCCC  TTTTTTTTTT
RRRRRRRR    EEEEEEEEEE    CCCCCCCC    SSSSSSSS  EEEEEEEEEE  LL          EEEEEEEEEE    CCCCCCCC  TTTTTTTTTT
RR      RR  EE          CC          SS          EE          LL          EE          CC              TT
RR      RR  EE          CC          SS          EE          LL          EE          CC              TT
RR      RR  EE          CC          SS          EE          LL          EE          CC              TT
RR      RR  EE          CC          SS          EE          LL          EE          CC              TT
RRRRRRRR    EEEEEEEE    CC            SSSSSS    EEEEEEEE    LL          EEEEEEEE    CC              TT
RRRRRRRR    EEEEEEEE    CC            SSSSSS    EEEEEEEE    LL          EEEEEEEE    CC              TT
RR  RR      EE          CC                  SS  EE          LL          EE          CC              TT
RR  RR      EE          CC                  SS  EE          LL          EE          CC              TT
RR    RR    EE          CC                  SS  EE          LL          EE          CC              TT
RR    RR    EE          CC                  SS  EE          LL          EE          CC              TT
RR      RR  EEEEEEEEEE    CCCCCCCC  SSSSSSSS    EEEEEEEEEE  LLLLLLLLLL  EEEEEEEEEE    CCCCCCCC      TT
RR      RR  EEEEEEEEEE    CCCCCCCC  SSSSSSSS    EEEEEEEEEE  LLLLLLLLLL  EEEEEEEEEE    CCCCCCCC      TT

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 RECSELECT
;     2          0002  0 (%TITLE 'Entry Validation'
;001 !SAR0293    0003  0 IDENT = 'V04-001') =
;     4-1        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 routine will determine if the previously read entry
;    37          0037  1 !       meets user specified selection criteria.
;    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:  January 1983
;    44          0044  1 !
;    45          0045  1 ! Modified by:
;    46          0046  1 !
;001 !SAR0293    0047  1 !       V04-001 SAR0293         Sharon A. Reynolds      1-Nov-1984
;002 !SAR0293    0048  1 !               Fixed a bug with /exclu=unknown.
;003 !SAR0293    0049  1 !
;    47          0050  1 !       V03-022 EAD0179         Elliott A. Drayton              6-Jul-1984
;    48          0051  1 !               Obtain LSTLUN value from SYECOM.
;    49          0052  1 !
;    50          0053  1 !       V03-023 SAR0274         Sharon A. Reynolds      19-Jun-1984
;    51          0054  1 !               - Added another check for device selection and entry
;    52          0055  1 !               selection combinations to fix a bug with
;    53          0056  1 !               /INC=(MF,VOLUME) and /INC=(TAPE,VOLUME).
;    54          0057  1 !
```

```
;    55          0058  1 !        V03-022 EAD0179         Elliott A. Drayton      23-May-1984
;    56          0059  1 !                Correct the passing of the address of device name
;    57          0060  1 !                in VERIFY_DEVICE.
;    58          0061  1 !
;    59          0062  1 !        V03-021 SAR0267         Sharon A. Reynolds      15-May-1984
;    60          0063  1 !                - Updated VERIFY_DEVICE to support longer device names.
;    61          0064  1 !                - Added check for unknown entry output to replace code
;    62          0065  1 !                that was previously removed.
;    63          0066  1 !
;    64          0067  1 !        V03-020 SAR0254         Sharon A. Reynolds      23-Apr-1984
;    65          0068  1 !                Added flag to /before check to stop execution when
;    66          0069  1 !                last entry found.
;    67          0070  1 !
;    68          0071  1 !        V03-019 EAD0151         Elliott A. Drayton      14-Apr-1984
;    69          0072  1 !                Fixed structure names in VERIFY_DEVICE.
;    70          0073  1 !
;    71          0074  1 !        V03-018 EAD0141         Elliott A. Drayton      12-Apr-1984
;    72          0075  1 !                Removed reference to EMBETDEF.
;    73          0076  1 !
;    74          0077  1 !        V03-017 SAR0248         Sharon A. Reynolds      10-Apr-1984
;    75          0078  1 !                Moved the unknown keyword tests to the verify entry
;    76          0079  1 !                routine so it would go through same tests as any
;    77          0080  1 !                other /include or /exclude entry selection.
;    78          0081  1 !
;    79          0082  1 !        V03-016 SAR0245         Sharon A. Reynolds      4-Apr-1984
;    80          0083  1 !                Added EMB$LOGMSP to device type entry table.
;    81          0084  1 !
;    82          0085  1 !        V03-015 EAD0119         Elliott A. Drayton      23-Mar-1984
;    83          0086  1 !                Remove support for /UNKNOWN qualifier and added support
;    84          0087  1 !                for the UNKNOWN keyword.
;    85          0088  1 !
;    86          0089  1 !        V03-014 EAD0115         Elliott A. Drayton       9-Mar-1984
;    87          0090  1 !                Removed emb_buf and syecom_buf.
;    88          0091  1 !
;    89          0092  1 !        V03-013 SAR0189         Sharon A. Reynolds,     13-Feb-1984
;    90          0093  1 !                - Added 'CS' device name support to device table search
;    91          0094  1 !                routine.
;    92          0095  1 !                - Added additional test for entry summary update.
;    93          0096  1 !
;    94          0097  1 !        V03-012 SAR0184         Sharon A. Reynolds,     17-Jan-1984
;    95          0098  1 !                - Fixed a bug in the output of the erf_unkentry message.
;    96          0099  1 !                - Added code to set the end value indicator when
;    97          0100  1 !                  the last selected entry (/entry) is found.
;    98          0101  1 !
;    99          0102  1 !        V03-011 SAR0181         Sharon A. Reynolds,     13-Dec-1983
;   100          0103  1 !                - Remove descriptor references.
;   101          0104  1 !                - Add device attention keyword support.
;   102          0105  1 !                - Add lm/sp entries to device errors entry list.
;   103          0106  1 !                - Add lm/sp entry check for bus class selections.
;   104          0107  1 !                - Removed logmessage keyword.
;   105          0108  1 !                - Add unsolicited_mscp keyword support.
;   106          0109  1 !                - Added incomplete entry message.
;   107          0110  1 !
;   108          0111  1 !        V03-010 SAR0176         Sharon A. Reynolds,     21-Nov-1983
;   109          0112  1 !                - Removed un-necessary check for outputting all
;   110          0113  1 !                  entries.
;   111          0114  1 !                - Changed reference to report type.
```

```
;  112     0115  1 !
;  113     0116  1 !       V03-009 SAR0152         Sharon A. Reynolds,     7-Oct-1983
;  114     0117  1 !               - Added code to output informational messages when
;  115     0118  1 !                 and unknown entry is encountered.
;  116     0119  1 !               - Added the code that counts intervening logmessage
;  117     0120  1 !                 logstatus entries.
;  118     0121  1 !               - Re-structured the /include and /exclude entry
;  119     0122  1 !                 checks to fix a bug.
;  120     0123  1 !               - Made /includ=disks/exclude=db1 a valid command.
;  121     0124  1 !
;  122     0125  1 !       V03-008 SAR0139         Sharon A. Reynolds,     20-Sep-1983
;  123     0126  1 !               Fixed a bug in mount/dismount output. Fixed an out
;  124     0127  1 !               of range loop.
;  125     0128  1 !
;  126     0129  1 !       V03-007 SAR0122         Sharon A. Reynolds,     23-Aug-1983
;  127     0130  1 !               Re-wrote translate_class routine for use with the
;  128     0131  1 !               permanent device tables.
;  129     0132  1 !
;  130     0133  1 !       V03-006 SAR0032         Sharon A. Reynolds,     2-Jun-1983
;  131     0134  1 !               Replaced emb_stuf with emb_buf definitions. Fixed bug
;  132     0135  1 !               in dc$_bus selection.
;  133     0136  1 !
;  134     0137  1 !       V03-005 SAR0029         Sharon A. Reynolds,     11-May-1983
;  135     0138  1 !               Removed support for logstatus keyword.
;  136     0139  1 !
;  137     0140  1 !       V03-004 SAR0013         Sharon A. Reynolds,     18-Apr-1983
;  138     0141  1 !               Deleted the log message and status message entries
;  139     0142  1 !               from the 'control' table. Added call to update
;  140     0143  1 !               entry summaries.
;  141     0144  1 !
;  142     0145  1 !       V03-003 SAR0003         Sharor A. Reynolds,     5-Apr-1983
;  143     0146  1 !               Removed the volume_output flag definition. Changed
;  144     0147  1 !               any references to volume_output flag so they refer
;  145     0148  1 !               to it from SYECOM.
;  146     0149  1 !
;  147     0150  1 !       V03-002 SAR0002         Sharon A. Reynolds,     5-Apr-1983
;  148     0151  1 !               Fixed /exclude selection bug.
;  149     0152  1 !
;  150     0153  1 !       V03-001 SAR0001         Sharon A. Reynolds,     29-Mar-1983
;  151     0154  1 !               Fixed /include='device name', volume mount/dismount
;  152     0155  1 !               selection problem.
;  153     0156  1 !
;  154     0157  1 !--
;  155     0158  1
;  156     0159  1
;  157     0160  1 !
;  158     0161  1 ! Required files
;  159     0162  1 !
;  160     0163  1 REQUIRE 'SRC$:ERFDEF.REQ' ;                 ! ERF defintions
;  161     0449  1 REQUIRE 'LIB$:PARSERDAT.R32' ;              ! ERF parser data definitions
;  162     0603  1 REQUIRE 'SRC$:RECSELDEF.REQ' ;              ! EMB, SYECOM, LOGMSG, LOGSTS, and
;  163     0734  1                                             !  VOLMOUNT field defintions
;  164     0735  1
;  165     0736  1 !
;  166     0737  1 ! Table of contents
;  167     0738  1 !
;  168     0739  1 !
```

```
;  169          0740  1 FORWARD ROUTINE
;  170          0741  1     Record_selected,                    ! Verify entry against selections
;  171          0742  1     Verify_entry,                       ! Verify the entry type
;  172          0743  1     Device_type_entry,                  ! Determine if it's a device type entry
;  173          0744  1     Verify_device_class,                ! Verify the device class
;  174          0745  1     Verify_device,                      ! Verify the device name
;  175          0746  1     Translate_class ;                   ! Translate device class to a name
;  176          0747  1
;  177          0748  1 !
;  178          0749  1 ! Declare external routines
;  179          0750  1 !
;  180          0751  1 EXTERNAL ROUTINE
;  181          0752  1     Exec_image,                         ! Execute an image
;  182          0753  1     Intervene_increment,
;  183          0754  1     Intervene_output,
;  184          0755  1     Search_queue: addressing_mode (general) , ! Search queue of devices selected
;  185          0756  1     Validate_packet;                    ! Is the packet validate for the cpu it was logged on.
;  186          0757  1
;  187          0758  1 !
;  188          0759  1 ! Declare external literals
;  189          0760  1 !
;  190          0761  1 EXTERNAL LITERAL
;  191          0762  1     Erf_incentry,
;  192          0763  1     Erf_unkclass,
;  193          0764  1     Erf_unkcpu,
;  194          0765  1     Erf_unkentry,
;  195          0766  1     Erf_unktype ;
;  196          0767  1
;  197          0768  1 !
;  198          0769  1 ! Declare external data.
;  199          0770  1 !
;  200          0771  1 EXTERNAL
;  201          0772  1     Class_dir:          REF $BBLOCK,
;  202          0773  1     Device_class,
;  203          0774  1     Device_type,
;  204          0775  1     Emb:                $BBLOCK PSECT (EMB),
;  205          0776  1     Exclude_flag,
;  206          0777  1     Exclude_mask:       REF $BBLOCK,
;  207          0778  1     Include_mask:       REF $BBLOCK,
;  208          0779  1     Option_flag:        REF $BBLOCK,
;  209          0780  1     Parser_data:        REF $BBLOCK,
;  210          0781  1     Processor_type,
;  211          0782  1     Summary_dispatcher_addr,
;  212          0783  1     Summary_flag:       REF $BBLOCK,
;  213          0784  1     Syecom:             $BBLOCK PSECT (SYECOM),
;  214          0785  1     Unknown_entry ;
;  215          0786  1
;  216          0787  1 !
;  217          0788  1 ! Declare literal definitions
;  218          0789  1 !
;  219          0790  1 LITERAL
;  220          0791  1     Incomplete_entry = 128 ;
;  221          0792  1
;  222          0793  1 !
;  223          0794  1 ! Own storage definitions
;  224          0795  1 !
;  225          0796  1 OWN
```

```
;  226     0797  1     Lstlun:              Long,
;  227     0798  1     Dev_selection_required: BYTE,
;  228     0799  1     Device_status:       BYTE,
;  229     0800  1     Dev_cls_status:      BYTE,
;  230     0801  1     Dev_type_entry_sts: BYTE,
;  231     0802  1     Entry_status:        BYTE,
;  232     0803  1     Validate_pkt_sts:    Initial (false),
;  233     0804  1     Bugchks:             VECTOR [3,byte,unsigned]  ! Bugcheck type entries
;  234     0805  1                          Initial (byte
;  235     0806  1                                   (EMB$K_CR,       ! Crash
;  236     0807  1                                   EMB$K_SBC,       ! System bugchecks
;  237     0808  1                                   EMB$K_UBC)),     ! User bugchecks
;  238     0809  1
;  239     0810  1     Control:             VECTOR [7,byte,unsigned] ! Control type entries
;  240     0811  1                          Initial (byte
;  241     0812  1                                   (EMB$K_CS,       ! Cold re-start
;  242     0813  1                                   EMB$K_NF,        ! New file created
;  243     0814  1                                   EMB$K_WS,        ! Warm re-start
;  244     0815  1                                   EMB$K_TS,        ! Time stamp
;  245     0816  1                                   EMB$K_SS,        ! System service message
;  246     0817  1                                   EMB$K_OM,        ! Operator message
;  247     0818  1                                   EMB$K_NM)),      ! Network message
;  248     0819  1
;  249     0820  1     Cpu:                 VECTOR [8,byte,unsigned] ! Cpu type entries
;  250     0821  1                          Initial (byte
;  251     0822  1                                   (EMB$K_AW,       ! Asynchronouw write error
;  252     0823  1                                   EMB$K_UBA,       ! Unibus adapter error
;  253     0824  1                                   EMB$K_MBA,       ! Massbus adapter error
;  254     0825  1                                   EMB$K_UI,        ! Undefined interrupt
;  255     0826  1                                   EMB$K_BE,        ! Bus error
;  256     0827  1                                   EMB$K_SA,        ! SBI alert
;  257     0828  1                                   EMB$K_SI,        ! 11/750 fault thru SBI vector
;  258     0829  1                                   EMB$K_UE)),      ! 11/730 unibus error
;  259     0830  1
;  260     0831  1     Dev_errors:          VECTOR [3,byte,unsigned] ! Device error entries
;  261     0832  1                          Initial (byte
;  262     0833  1                                   (EMB$K_DE,       ! Device Errors
;  263     0834  1                                   EMB$K_SP,        ! Logstatus entries (mscp)
;  264     0835  1                                   EMB$K_LM)),      ! Logmessage entries (mscp)
;  265     0836  1
;  266     0837  1     Memorys:             VECTOR [2,byte,unsigned] ! Memory entries
;  267     0838  1                          Initial (byte
;  268     0839  1                                   (EMB$K_SE,       ! Soft ECC error
;  269     0840  1                                   EMB$K_HE)),      ! Hard ECC error
;  270     0841  1
;  271     0842  1     Volume:              VECTOR [2,byte,unsigned]         ! Volume change entries
;  272     0843  1                          Initial (byte
;  273     0844  1                                   (EMB$K_VM,       ! Volume mounts
;  274     0845  1                                   EMB$K_VD)) ;     ! Volume dismounts
;  275     0846  1
```

```
;  277          0847  1 GLOBAL ROUTINE RECORD_SELECTED =
;  278          0848  2 Begin
;  279          0849  2
;  280          0850  2 !++
;  281          0851  2 !
;  282          0852  2 ! Functional Description:
;  283          0853  2 !
;  284          0854  2 !       This routine will determine what selection qualifiers are
;  285          0855  2 !       specified and match the appropriate fields in the current
;  286          0856  2 !       entry against the selections. It return TRUE if the
;  287          0857  2 !       current entry matches or return FALSE if the current entry
;  288          0858  2 !       does NOT match.
;  289          0859  2 !
;  290          0860  2 ! Calling sequence:
;  291          0861  2 !
;  292          0862  2 !       RECORD_SELECTED ()
;  293          0863  2 !
;  294          0864  2 ! Input parameters:
;  295          0865  2 !
;  296          0866  2 !       None
;  297          0867  2 !
;  298          0868  2 ! Output parameters:
;  299          0869  2 !
;  300          0870  2 !       None
;  301          0871  2 !
;  302          0872  2 !--
;  303          0873  2
;  304          0874  2 LOCAL
;  305          0875  2     Include_status:     BYTE
;  306          0876  2                         Initial (true),
;  307          0877  2     Exclude_status:     BYTE
;  308          0878  2                         Initial (true) ;
;  309          0879  2
;  310          0880  2 lstlun = .syecom [sye$l_lstlun];
;  311          0881  2
;  312          0882  2 !
;  313          0883  2 ! Validate the packet for entry/cpu type and device class/type.
;  314          0884  2 !
;  315          0885  3 If NOT (VALIDATE_PACKET ())
;  316          0886  2 Then
;  317          0887  2     Unknown_entry = true
;  318          0888  2 Else
;  319          0889  2     Unknown_entry = false ;
;  320          0890  2
;  321          0891  2 !
;  322          0892  2 ! Determine if /summary selected and update that entry summary
;  323          0893  2 ! information.
;  324          0894  2 !
;  325          0895  3 If (.option_flag[opt$v_summary_qual] AND
;  326          0896  4     (.summary_flag[sum$v_entry] OR
;  327          0897  4      .summary_flag[sum$v_all_summ] OR
;  328          0898  3      .summary_flag[sum$v_histogram]))
;  329          0899  2 Then
;  330          0900  2     Exec_image (summary_dispatcher_addr,lstlun,%REF(entry_summ_upd)) ;
;  331          0901  2
;  332          0902  2 !
;  333          0903  2 ! If incomplete entry report the error.
```

```
;  334     0904  2 !
;  335     0905  3 If ((NOT .syecom[sye$b_valid_entry]) AND
;  336     0906  3     (.emb[emb$w_hd_entry] GEQ incomplete_entry))
;  337     0907  2 Then
;  338     0908  3     Begin
;  339     0909  3     Signal (erf_incentry, 1, .emb[emb$w_hd_entry]);
;  340     0910  3     Return false;
;  341     0911  2     End;
;  342     0912  2
;  343     0913  2 !
;  344     0914  2 ! Determine whether the volume mounts/dismounts should be output or just
;  345     0915  2 ! label information saved from the entry.
;  346     0916  2 !
;  347     0917  3 If (.exclude_mask[exc$v_volume] AND
;  348     0918  4     (.include_mask[inc$v_device_select] OR
;  349     0919  4     .include_mask[inc$v_dev_class_select] OR
;  350     0920  4     .include_mask[inc$v_dev_attentions] OR
;  351     0921  4     .include_mask[inc$v_dev_errors] OR
;  352     0922  2     .include_mask[inc$v_dev_timeouts])) AND
;  353     0923  3     (NOT .include_mask[inc$v_volume] OR
;  354     0924  3     NOT .option_flag[opt$v_output_all])
;  355     0925  2 Then
;  356     0926  2         !
;  357     0927  2         ! Indicate that volume mount/dismount entries
;  358     0928  2         ! should not be output.
;  359     0929  2         !
;  360     0930  2         Syecom[sye$b_volume_output] = false
;  361     0931  2 Else
;  362     0932  2         Syecom[sye$b_volume_output] = true ;
;  363     0933  2
;  364     0934  2 !
;  365     0935  2 ! Determine if the /ENTRY qualifier was specified.
;  366     0936  2 !
;  367     0937  2 If .option_flag[opt$v_entry_qual]
;  368     0938  2 Then
;  369     0939  2         !
;  370     0940  2         ! /Entry specified, get the address of the entry selection
;  371     0941  2         ! data and determine if the number of this entry
;  372     0942  2         ! is within the selected range.
;  373     0943  2         !
;  374     0944  3         Begin
;  375     0945  3         If .syecom[sye$l_reccnt] LSSU .parser_data[erl$l_end_entry]
;  376     0946  3         Then
;  377     0947  3             !
;  378     0948  3             ! This entry should be within the selected range, ensure
;  379     0949  3             ! the entry number is greater than the starting entry selection.
;  380     0950  3             !
;  381     0951  4             Begin
;  382     0952  5             If NOT (.syecom[sye$l_reccnt] GEQU .parser_data[erl$l_start_entry])
;  383     0953  4             Then
;  384     0954  4                 !
;  385     0955  4                 ! Entry is NOT within the selected range, return to calling
;  386     0956  4                 ! routine.
;  387     0957  4                 !
;  388     0958  4                 Return false ;
;  389     0959  4             End
;  390     0960  3         Else
```

```
;  391          0961  3          !
;  392          0962  3          ! Entry is NOT within the selected range, return to calling
;  393          0963  3          ! routine.
;  394          0964  3          !
;  395          0965  4          Begin
;  396          0966  4          If .syecom[sye$l_reccnt] GEQU .parser_data[erl$l_end_entry]
;  397          0967  4          Then
;  398          0968  4              !
;  399          0969  4              ! Indicate that last selected entry was found.
;  400          0970  4              !
;  401          0971  4              Syecom[sye$b_end_value] = true ;
;  402          0972  4
;  403          0973  4          Return true ;
;  404          0974  3          End ;
;  405          0975  2      End ;
;  406          0976  2
;  407          0977  2 !
;  408          0978  2 ! Determine if the /BEFORE qualifier was specified.
;  409          0979  2 !
;  410          0980  2 If .option_flag[opt$v_before_qual]
;  411          0981  2 Then
;  412          0982  2     !
;  413          0983  2     ! Determine if the date/time that this entry was recorded falls
;  414          0984  2     ! within the range of selected date/times.
;  415          0985  2     !
;  416          0986  3     Begin
;  417        P 0987  3     If COMPARE_QUAD(emb[emb$q_hd_time],GEQU,
;  418          0988  4                     parser_data[erl$q_end_date])
;  419          0989  3     Then
;  420          0990  3         !
;  421          0991  3         ! This entry is NOT within the selected date/time range,
;  422          0992  3         ! return to the calling routine.
;  423          0993  3         !
;  424          0994  4         Begin
;  425          0995  4         Syecom[sye$b_end_value] = true ,
;  426          0996  4         Return true ;
;  427          0997  3         End ;
;  428          0998  2     End ;
;  429          0999  2
;  430          1000  2 !
;  431          1001  2 ! Determine if the /SINCE qualifier was specified.
;  432          1002  2 !
;  433          1003  2 If .option_flag[opt$v_since_qual]
;  434          1004  2 Then
;  435          1005  2     !
;  436          1006  2     ! Ensure that the entry date/time is greater than the starting
;  437          1007  2     ! time/date selection.
;  438          1008  2     !
;  439          1009  3     Begin
;  440        P 1010  3     If NOT COMPARE_QUAD(emb[emb$q_hd_time],GEQU,
;  441          1011  4                         parser_data[erl$q_start_date])
;  442          1012  3     Then
;  443          1013  3         !
;  444          1014  3         ! The entry does NOT meet that selection criteria for date/time,
;  445          1015  3         ! return to the calling routine.
;  446          1016  3         !
;  447          1017  3         Return false ;
```

```
;  448          1018  2     End ;
;  449          1019  2
;  450          1020  2 !
;  451          1021  2 ! Determine if the /SID_REGISTER qualifier was specified.
;  452          1022  2 !
;  453          1023  2 If .option_flag[opt$v_sid_reg_qual]
;  454          1024  2 Then
;  455          1025  2     !
;  456          1026  2     ! Determine if the entry sid matches the selected sid.
;  457          1027  2     !
;  458          1028  3     Begin
;  459          1029  3     If NOT .parser_data[erl$l_sid_selection] EQLU .emb[emb$l_hd_sid]
;  460          1030  3     Then
;  461          1031  3         !
;  462          1032  3         ! Entry sid does NOT match selected sid, return to calling
;  463          1033  3         ! routine.
;  464          1034  3         !
;  465          1035  3         Return false ;
;  466          1036  2     End ;
;  467          1037  2
;  468          1038  2 Device_status = false ;
;  469          1039  2 Dev_cls_status = false ;
;  470          1040  2 Entry_status = false ;
;  471          1041  2
;  472          1042  2 Dev_type_entry_sts = DEVICE_TYPE_ENTRY () ;
;  473          1043  2
;  474          1044  2 If .option_flag[opt$v_include_qual]
;  475          1045  2 Then
;  476          1046  3     Begin
;  477          1047  3     Exclude_flag = false ;
;  478          1048  3
;  479          1049  3     If .dev_type_entry_sts OR
;  480          1050  3         (.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  481          1051  4         (.emb[emb$w_hd_entry] EQLU EMB$K_VD)
;  482          1052  3     Then
;  483          1053  4         Begin
;  484          1054  4         If .include_mask[inc$v_device_select]
;  485          1055  4         Then
;  486          1056  5             Begin
;  487          1057  5             If VERIFY_DEVICE ()
;  488          1058  5             Then
;  489          1059  5                 Device_status = true
;  490          1060  5             Else
;  491          1061  5                 Device_status = false ;
;  492          1062  4             End ;
;  493          1063  4
;  494          1064  4         If .include_mask[inc$v_dev_class_select]
;  495          1065  4         Then
;  496          1066  5             Begin
;  497          1067  5             If VERIFY_DEVICE_CLASS ()
;  498          1068  5             Then
;  499          1069  5                 Dev_cls_status = true
;  500          1070  5             Else
;  501          1071  5                 Dev_cls_status = false ;
;  502          1072  4             End ;
;  503          1073  3         End ;
;  504          1074  3
```

```
;  505          1075  3     If .include_mask[inc$v_entry_select]
;  506          1076  3     Then
;  507          1077  4         Begin
;  508          1078  4         If VERIFY_ENTRY ()
;  509          1079  4         Then
;  510          1080  4             Entry_status = true
;  511          1081  4         Else
;  512          1082  4             Entry_status = false ;
;  513          1083  3         End ;
;  514          1084  3
;  515          1085  3
;  516          1086  4     If (.include_mask[inc$v_device_select] AND
;  517          1087  3         .dev_type_entry_sts AND .device_status) OR
;  518          1088  3
;  519          1089  4         (.include_mask[inc$v_dev_class_select] AND
;  520          1090  3         .dev_type_entry_sts AND .dev_class_status) OR
;  521          1091  3
;  522          1092  4         (.include_mask[inc$v_entry_select] AND .entry_status)
;  523          1093  3     Then
;  524          1094  3         Include_status = true
;  525          1095  3     Else
;  526          1096  3         Include_status = false ;
;  527          1097  3
;  528          1098  3
;  529          1099  3     If .include_mask[inc$v_device_select] AND
;  530          1100  3         .include_mask[inc$v_entry_select]
;  531          1101  3     Then
;  532          1102  4         Begin
;  533          1103  4         Include_status = false ;
;  534          1104  4
;  535          1105  4         If .dev_selection_required
;  536          1106  4         Then
;  537          1107  5             Begin
;  538          1108  5             If (.entry_status AND .device_status) OR
;  539          1109  6                 (.dev_type_entry_sts AND .device_status)
;  540          1110  5             Then
;  541          1111  5                 Include_status = true ;
;  542          1112  5             End
;  543          1113  4         Else
;  544          1114  5             Begin
;  545          1115  5             If .dev_type_entry_sts AND .device_status
;  546          1116  5             Then
;  547          1117  6                 Begin
;  548          1118  6                 Include_status = true ;
;  549          1119  6                 End
;  550          1120  5             Else
;  551          1121  6                 Begin
;  552          1122  7                 If (NOT .dev_type_entry_sts AND .entry_status)
;  553          1123  6                 Then
;  554          1124  6                     Include_status = true ;
;  555          1125  5                 End ;
;  556          1126  4             End ;
;  557          1127  3         End ;
;  558          1128  3
;  559          1129  3     If .include_mask[inc$v_dev_class_select] AND
;  560          1130  3         .include_mask[inc$v_entry_select]
;  561          1131  3     Then
```

```
;  562          1132  4          Begin
;  563          1133  4          Include_status = false ;
;  564          1134  4
;  565          1135  4          If .dev_selection_required
;  566          1136  4          Then
;  567          1137  5              Begin
;  568          1138  5              If (.entry_status AND .dev_cls_status) OR
;  569          1139  6                  (.dev_type_entry_sts AND .dev_cls_status)
;  570          1140  5              Then
;  571          1141  5                  Include_status = true ;
;  572          1142  5              End
;  573          1143  4          Else
;  574          1144  5              Begin
;  575          1145  5              If .dev_type_entry_sts AND .dev_cls_status
;  576          1146  5              Then
;  577          1147  6                  Begin
;  578          1148  6                  Include_status = true ;
;  579          1149  6                  End
;  580          1150  5              Else
;  581          1151  6                  Begin
;  582          1152  7                  If (NOT .dev_type_entry_sts AND .entry_status)
;  583          1153  6                  Then
;  584          1154  6                      Include_status = true ;
;  585          1155  5                  End ;
;  586          1156  4              End ;
;  587          1157  3          End ;
;  588          1158  3
;  589          1159  2      End ;
;  590          1160  2
;  591          1161  2 !
;  592          1162  2 !If not /include option then include_status = false
;  593          1163  2 !
;  594          1164  2
;  595          1165  2 If .option_flag[opt$v_exclude_qual]
;  596          1166  2 Then
;  597          1167  3     Begin
;  598          1168  3     Exclude_flag = true ;
;  599          1169  3
;  600          1170  3     If .dev_type_entry_sts OR
;  601          1171  3         (.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  602          1172  4         (.emb[emb$w_hd_entry] EQLU EMB$K_VD)
;  603          1173  3     Then
;  604          1174  4         Begin
;  605          1175  4         If .exclude_mask[exc$v_device_select]
;  606          1176  4         Then
;  607          1177  5             Begin
;  608          1178  5             If VERIFY_DEVICE ()
;  609          1179  5             Then
;  610          1180  5                 Device_status = true
;  611          1181  5             Else
;  612          1182  5                 Device_status = false ;
;  613          1183  4             End ;
;  614          1184  4
;  615          1185  4         If .exclude_mask[exc$v_dev_class_select]
;  616          1186  4         Then
;  617          1187  5             Begin
;  618          1188  5             If VERIFY_DEVICE_CLASS ()
```

```
;  619     1189  5              Then
;  620     1190  5                  Dev_cls_status = true
;  621     1191  5              Else
;  622     1192  5                  Dev_cls_status = false ;
;  623     1193  4              End ;
;  624     1194  3          End ;
;  625     1195  3
;  626     1196  3      If .exclude_mask[exc$v_entry_select]
;  627     1197  3      Then
;  628     1198  4          Begin
;  629     1199  4          If VERIFY_ENTRY ()
;  630     1200  4          Then
;  631     1201  4              Entry_status = true
;  632     1202  4          Else
;  633     1203  4              Entry_status = false ;
;  634     1204  3          End ;
;  635     1205  3
;  636     1206  4      If (.exclude_mask[exc$v_device_select] AND
;  637     1207  3          .dev_type_entry_sts AND .device_status) OR
;  638     1208  3
;  639     1209  4          (.exclude_mask[exc$v_dev_class_select] AND
;  640     1210  3          .dev_type_entry_sts AND .dev_cls_status) OR
;  641     1211  3
;  642     1212  4          (.exclude_mask[exc$v_entry_select] AND .entry_status)
;  643     1213  3      Then
;  644     1214  3          Exclude_status = false
;  645     1215  3      Else
;  646     1216  3          Exclude_status = true ;
;  647     1217  3
;  648     1218  3
;  649     1219  3      If .exclude_mask[exc$v_device_select] AND
;  650     1220  3          .exclude_mask[exc$v_entry_select]
;  651     1221  3      Then
;  652     1222  4          Begin
;  653     1223  4          Exclude_status = true ;
;  654     1224  4
;  655     1225  4          If .dev_selection_required
;  656     1226  4          Then
;  657     1227  5              Begin
;  658     1228  5              If (.entry_status AND .device_status) OR
;  659     1229  6                  (.dev_type_entry_sts AND .device_status)
;  660     1230  5              Then
;  661     1231  5                  Exclude_status = false ;
;  662     1232  5              End
;  663     1233  4          Else
;  664     1234  5              Begin
;  665     1235  5              If .dev_type_entry_sts AND .device_status
;  666     1236  5              Then
;  667     1237  6                  Begin
;  668     1238  6                  Exclude_status = false ;
;  669     1239  6                  End
;  670     1240  5              Else
;  671     1241  6                  Begin
;  672     1242  7                  If (NOT .dev_type_entry_sts AND .entry_status)
;  673     1243  6                  Then
;  674     1244  6                      Exclude_status = false ;
;  675     1245  5                  End ;
```

```
;  676     1246  4              End ;
;  677     1247  3          End ;
;  678     1248  3
;  679     1249  3      If .exclude_mask[exc$v_dev_class_select] AND
;  680     1250  3         .exclude_mask[exc$v_entry_select]
;  681     1251  3      Then
;  682     1252  4          Begin
;  683     1253  4          Exclude_status = true ;
;  684     1254  4
;  685     1255  4          If .dev_selection_required
;  686     1256  4          Then
;  687     1257  5              Begin
;  688     1258  5              If (.entry_status AND .dev_cls_status) OR
;  689     1259  6                  (.dev_type_entry_sts AND .dev_cls_status)
;  690     1260  5              Then
;  691     1261  5                  Exclude_status = false ;
;  692     1262  5              End
;  693     1263  4          Else
;  694     1264  5              Begin
;  695     1265  5              If .dev_type_entry_sts AND .dev_cls_status
;  696     1266  5              Then
;  697     1267  6                  Begin
;  698     1268  6                  Exclude_status = false ;
;  699     1269  6                  End
;  700     1270  5              Else
;  701     1271  6                  Begin
;  702     1272  7                  If (NOT .dev_type_entry_sts AND .entry_status)
;  703     1273  6                  Then
;  704     1274  6                      Exclude_status = false ;
;  705     1275  5                  End ;
;  706     1276  4              End ;
;  707     1277  3          End ;
;  708     1278  3
;  709     1279  2      End ;       ! of /exclude processing
;  710     1280  2 !
;  711     1281  2 ! IF /exclude option match, exclude_status = false.
;  712     1282  2 !
;  713     1283  2
;  714     1284  2
;  715     1285  2 !
;  716     1286  2 ! Determine whether to count logmessage/logstatus entries.
;  717     1287  2 !
;  718     1288  3 If ( (.include_status) AND (.exclude_status) AND
;  719     1289  3      (.parser_data[erl$b_rpt_type] EQ[ full_rep) )
;  720     1290  2 Then
;  721     1291  2     !
;  722     1292  2     ! Determine if it was a logmessage/logstatus entry.
;  723     1293  2     !
;  724     1294  3     Begin
;  725     1295  3     If (.emb[emb$w_hd_entry] EQLU EMB$C_SP) OR
;  726     1296  4         (.emb[emb$w_hd_entry] EQLU EMB$C_LM)
;  727     1297  3     Then
;  728     1298  3         !
;  729     1299  3         ! Count the number of logmessage/logstatus entries
;  730     1300  3         ! that might be skipped.
;  731     1301  3         !
;  732     1302  3         INTERVENE_INCREMENT (lstlum)
```

```
;   733          1303  3     Else
;   734          1304  3         !
;   735          1305  3         ! Determine whether to output the logstatus/logmessage
;   736          1306  3         ! intervening message and if necessary output it.
;   737          1307  3         !
;   738          1308  3         INTERVENE_OUTPUT (lstlun) ;
;   739          1309  2     End ;
;   740          1310  2
;   741          1311  2 !
;   742          1312  2 ! Determine if the entry met the selection criteria.
;   743          1313  2 !
;   744          1314  2 ! Determine if this is an unknown entry.
;   745          1315  2 !
;   746          1316  2 If .unknown_entry
;   747          1317  2 Then
;001 !SAR0293    1318  2     ! Determine whether this entry should be output.
;002 !SAR0293    1319  2     !
;003 !SAR0293    1320  3     Begin
;004 !SAR0293    1321  3     If .exclude_mask[exc$v_unknown_entry]
;005 !SAR0293    1322  3     Then
;006 !SAR0293    1323  3         ! Indicate that this entry should not be output.
;007 !SAR0293    1324  3         !
;008 !SAR0293    1325  3         Return false
;009 !SAR0293    1326  3     Else
;010 !SAR0293    1327  3         ! Indicate that this entry should be output.
;011 !SAR0293    1328  3         !
;012 !SAR0293    1329  3         Return true ;
;013 !SAR0293    1330  2     End ;
;014 !SAR0293    1331  2
;   752-4        1332  2
;   753          1333  2 If (NOT .include_status) OR
;   754          1334  3     (NOT .exclude_status)
;   755          1335  2 Then
;   756          1336  2     !
;   757          1337  2     ! Indicate that the entry should not be output by
;   758          1338  2     ! returning to the calling routine with a false value.
;   759          1339  2     !
;   760          1340  2     Return false ;
;   761          1341  2
;   762          1342  2 !
;   763          1343  2 ! Indicate that the entry should be output by
;   764          1344  2 ! returning to the calling routine with a true value.
;   765          1345  2 !
;   766          1346  2 Return true ;
;   767          1347  2
;   768          1348  1 End ;    ! Routine


                                                .TITLE  RECSELECT Entry Validation
                                                .IDENT  \V04-001\

                                                .PSECT  $OWN$,NOEXE,  PIC,2

                                    00000 LSTLUN: .BLKB   4
                                    00004 DEV_SELECTION_REQUIRED:
                                                .BLKB   1
                                    00005 DEVICE_STATUS:
```

```
                                                        .BLKB   1
                                                00006 DEV_CLS_STATUS:
                                                        .BLKB   1
                                                00007 DEV_TYPE_ENTRY_STS:
                                                        .BLKB   1
                                                00008 ENTRY_STATUS:
                                                        .BLKB   1
                                                00009   .BLKB   3
                                      00000000  0000C VALIDATE_PKT_STS:
                                                        .LONG   0
                                   70 28 25     00010 BUGCHKS:.BYTE   37, 40, 112                                ;
                                                0001:3  .BLKB   1
                                2A 29 27 26 24 23 20    00014 CONTROL:.BYTE   32, 35, 36, 38, 39, 41, 42        ;
                                                0001B   .BLKB   1
                             0B 0A 05 04 61 0C 09 07    0001C CPU:    .BYTE   7, 9, 12, 97, 4, 5, 10, 11        ;
                                   64 63 01     00024 DEV_ERRORS:
                                                        .BYTE   1, 99, 100                              ;
                                                00027   .BLKB   1
                                      08 06     00028 MEMORYS:.BYTE   6, 8                                ;
                                                0002A   .BLKB   2
                                      41 40     0002C VOLUME: .BYTE   64, 65                              ;

                                                        .EXTRN  EXEC_IMAGE, INTERVENE_INCREMENT
                                                        .EXTRN  INTERVENE_OUTPUT
                                                        .EXTRN  SEARCH_QUEUE, VALIDATE_PACKET
                                                        .EXTRN  ERF_INCENTRY, ERF_UNKCLASS
                                                        .EXTRN  ERF_UNKCPU, ERF
                                                        .EXTRN  ERF_UNKTYPE, CLAS
                                                        .EXTRN  DEVICE_CLASS, DEVI..
                                                        .EXTRN  EMB, EXCLUDE_FLAG
                                                        .EXTRN  EXCLUDE_MASK, INCLUDE_MA.
                                                        .EXTRN  OPTION_FLAG, PARSER_DATA
                                                        .EXTRN  PROCESSOR_TYPE, SUMMARY_DIS. . ADDR
                                                        .EXTRN  SUMMARY_FLAG, SYECOM
                                                        .EXTRN  UNKNOWN_ENTRY

                                                        .PSECT  $CODE,NOWRT,  PIC,2

                                   0FFC 00000           .ENTRY  RECORD_SELECTED, Save R2,R3,R4,R5,R6,R7,R8,-; 0847
                                                                R9,R10,R11
                                5B 00000000G   00  9E 00002         MOVAB   PARSER_DATA, R11                ;
                                5A 00000000G   00  9E 00009         MOVAB   OPTION_FLAG, R10                ;
                                59 00000000G   00  9E 00010         MOVAB   SYECOM+24, R9                   ;
                                58 00000000G   00  9E 00017         MOVAB   INCLUDE_MASK, R8                ;
                                57 00000000G   00  9E 0001E         MOVAB   EXCLUDE_MASK, R7                ;
                                56 00000000G   00  9E 00025         MOVAB   EMB+4, R6                       ;
                                55 00000000'   00  9E 0002C         MOVAB   ENTRY_STATUS, R5                ;
                                5E             04  C2 00033         SUBL2   #4, SP                          ;
                                54             01  90 00036         MOVB    #1, INCLUDE_STATUS              ; 0848
                                53             01  90 00039         MOVB    #1, EXCLUDE_STATUS              ;
                             F8 A5       0F    A9  D0 0003C         MOVL    SYECOM+39, CSTLUN               ; 0880
                00000000G    00             00  FB 00041         CALLS   #0, VALIDATE_PACKET             ; 0885
                             09             50  E8 00048         BLBS    R0, 1$                          ;
                00000000G    00             01  D0 0004B         MOVL    #1, UNKNOWN_ENTRY               ; 0887
                                            06  11 00052         BRB     2$                              ;
                                00000000G   00  D4 00054 1$:      CLRL    UNKNOWN_ENTRY                   ; 0889
                             50             6A  D0 0005A 2$:      MOVL    OPTION_FLAG, R0                 ; 0895
```

```
                27              60                  0E  E1 0005D          BBC     #14, (R0), 4$                                   ;
                                50 00000000G        00  D0 00061          MOVL    SUMMARY_FLAG, R0                                ; 0896
                07              60                  02  E0 00068          BBS     #2, (R0), 3$                                    ;
                                04                  60  E8 0006C          BLBS    (R0), 3$                                        ; 0897
                15              60                  05  E1 0006F          BBC     #5, (R0), 4$                                    ; 0898
                                6E                  05  D0 00073 3$:      MOVL    #5, (SP)                                        ; 0900
                                                    5E  DD 00076          PUSHL   SP
                                              F8    A5  9F 00078          PUSHAB  LSTLUN
                                    00000000G       00  9F 0007B          PUSHAB  SUMMARY_DISPATCHER_ADDR
                    00000000G   00                  03  FB 00081          CALLS   #3, EXEC_IMAGE
                                1C            03    A9  E8 00088 4$:      BLBS    SYECOM+27, 6$                                   ; 0905
                        0080    8F                  66  B1 0008C          CMPW    EMB+4, #128                                     ; 0906
                                                    15  1F 00091          BLSSU   6$
                                7E                  66  3C 00093          MOVZWL  EMB+4, -(SP)                                    ; 0909
                                                    01  DD 00096          PUSHL   #1
                                    00000000G       8F  DD 00098          PUSHL   #ERF_INCENTRY
                    00000000G   00                  03  FB 0009E          CALLS   #3, LIB$SIGNAL
                                                02F2    31  000A5 5$:      BRW     70$                                             ; 0910
                                50                  67  D0 000A8 6$:      MOVL    EXCLUDE_MASK, R0                                ; 0917
                29              60                  12  E1 000AB          BBC     #18, (R0), 9$
                                50                  68  D0 000AF          MOVL    INCLUDE_MASK, R0                                ; 0918
                10              60                  14  E0 000B2          BBS     #20, (R0), 7$
                0C              60                  15  E0 000B6          BBS     #21, (R0), 7$                                   ; 0919
                08              60                  09  E0 000BA          BBS     #9, (R0), 7$                                    ; 0920
                04              60                  0D  E0 000BE          BBS     #13, (R0), 7$                                   ; 0921
                                12            02    A0  E9 000C2          BLBC    2(R0), 9$                                       ; 0922
                                50                  68  D0 000C6 7$:      MOVL    INCLUDE_MASK, R0                                ; 0923
                07              60                  12  E1 000C9          BBC     #18, (R0), 8$
                                50                  6A  D0 000CD          MOVL    OPTION_FLAG, R0                                 ; 0924
                                                    60  B5 000D0          TSTW    (R0)
                                                    04  19 000D2          BLSS    9$
                                                    69  94 000D4 8$:      CLRB    SYECOM+24                                       ; 0930
                                                    03  11 000D6          BRB     10$
                                69                  01  90 000D8 9$:      MOVB    #1, SYECOM+24                                   ; 0932
                                52                  6A  D0 000DB 10$:     MOVL    OPTION_FLAG, R2                                 ; 0937
                13              62                  03  E1 000DE          BBC     #3, (R2), 8$
                                51            E8    A9  D0 000E2          MOVL    SYECOM, R1                                      ; 0945
                                50                  6B  D0 000E6          MOVL    PARSER_DATA, R0
                        19      A0                  51  D1 000E9          CMPL    R1, 25(R0)
                                                    1D  1E 000ED          BGEQU   13$
                        15      A0                  51  D1 000EF          CMPL    R1, 21(R0)                                      ; 0952
                                                    B0  1F 000F3          BLSSU   5$
                                1B                  62  E9 000F5 11$:     BLBC    (R2), 14$                                       ; 0980
                50              6B                  05  C1 000F8          ADDL3   #5, PARSER_DATA, R0                             ; 0988
                                51            06    A6  D0 000FC          MOVL    A+4, R1
                        04      A0                  51  D1 00100          CMPL    R1, 4(R0)
                                                    04  12 00104          BNEQ    12$
                                60            02    A6  D1 00106          CMPL    A, (R0)
                                                    07  1F 0010A 12$:     BLSSU   14$
                        06      A9                  01  90 0010C 13$:     MOVB    #1, SYECOM+30                                   ; 0995
                                                02B3    31  00110          BRW     69$                                             ; 0996
                18              62                  0D  E1 00113 14$:     BBC     #13, (R2), 16$                                  ; 1003
                50              6B                  0D  C1 00117          ADDL3   #13, PARSER_DATA, R0                            ; 1011
                                51            06    A6  D0 0011B          MOVL    A+4, R1
                        04      A0                  51  D1 0011F          CMPL    R1, 4(R0)
                                                    08  12 00123          BNEQ    15$
                                60            02    A6  D1 00125          CMPL    A, (R0)
```

```
                                    12  1F 00129          BLSSU   17$                                          ;
                                    02  11 0012B          BRB     16$                                          ;
                                    0E  1F 0012D 15$:     BLSSU   17$                                          ;
                 0D         62      0C  E1 0012F 16$:     BBC     #12, (R2), 18$                               ; 1023
                            50      6B  D0 00133          MOVL    PARSER_DATA, R0                              ; 1029
                    FC  A6      01  A0  D1 00136          CMPL    1(R0), EMB                                   ;
                            03      03  13 0013B          BEQL    18$                                          ;
                                025A  31 0013D 17$:       BRW     70$                                          ;
                        FD  A5      B4 00140 18$:         CLRW    DEVICE_STATUS                                ; 1038
                            65      94 00143              CLRB    ENTRY_STATUS                                 ; 1040
              00000000V     00      00 FB 00145           CALLS   #0, DEVICE_TYPE_ENTRY                        ; 1042
                    FF  A5          50 90 0014C           MOVB    R0, DEV_TYPE_ENTRY_STS                       ;
                            50      6A D0 00150           MOVL    OPTION_FLAG, R0                              ; 1044
                 03         60      06 E0 00153           BBS     #6, (R0), 19$                                ;
                                00F3  31 00157            BRW     41$                                          ;
                        00000000G   00  D4 0015A 19$:     CLRL    EXCLUDE_FLAG                                 ; 1047
                            0E  FF  A5 E8 00160           BLBS    DEV_TYPE_ENTRY_STS, 20$                      ; 1049
                      0040  8F      66 B1 00164           CMPW    EMB+4, #64                                   ; 1050
                                    07 13 00169           BEQL    20$                                          ;
                      0041  8F      66 B1 0016B           CMPW    EMB+4, #65                                   ; 1051
                                    34 12 00170           BNEQ    24$                                          ;
                            50      68 D0 00172 20$:      MOVL    INCLUDE_MASK, R0                             ; 1054
                 13         60      14 E1 00175           BBC     #20, (R0), 22$                               ;
              00000000V     00      00 FB 00179           CALLS   #0, VERIFY_DEVICE                            ; 1057
                            06      50 E9 00180           BLBC    R0, 21$                                      ;
                    FD  A5          01 90 00183           MOVB    #1, DEVICE_STATUS                            ; 1059
                                    03 11 00187           BRB     22$                                          ;
                        FD  A5      94 00189 21$:         CLRB    DEVICE_STATUS                                ; 1061
                            50      68 D0 0018C 22$:      MOVL    INCLUDE_MASK, R0                             ; 1064
                 13         60      15 E1 0018F           BBC     #21, (R0), 24$                               ;
              00000000V     00      00 FB 00193           CALLS   #0, VERIFY_DEVICE_CLASS                      ; 1067
                            06      50 E9 0019A           BLBC    R0, 23$                                      ;
                    FE  A5          01 90 0019D           MOVB    #1, DEV_CLS_STATUS                           ; 1069
                                    03 11 001A1           BRB     24$                                          ;
                        FE  A5      94 001A3 23$:         CLRB    DEV_CLS_STATUS                               ; 1071
                            50      68 D0 001A6 24$:      MOVL    INCLUDE_MASK, R0                             ; 1075
                 11         60      16 E1 001A9           BBC     #22, (R0), 26$                               ;
              00000000V     00      00 FB 001AD           CALLS   #0, VERIFY_ENTRY                             ; 1078
                            05      50 E9 001B4           BLBC    R0, 25$                                      ;
                            65      01 90 001B7           MOVB    #1, ENTRY_STATUS                             ; 1080
                                    02 11 001BA           BRB     26$                                          ;
                                    65 94 001BC 25$:      CLRB    ENTRY_STATUS                                 ; 1082
                            50      68 D0 001BE 26$:      MOVL    INCLUDE_MASK, R0                             ; 1086
                 08         60      14 E1 001C1           BBC     #20, (R0), 27$                               ;
                            04  FF  A5 E9 001C5           BLBC    DEV_TYPE_ENTRY_STS, 27$                      ; 1087
                            13  FD  A5 E8 001C9           BLBS    DEVICE_STATUS, 29$                           ;
                 08         60      15 E1 001CD 27$:      BBC     #21, (R0), 28$                               ; 1089
                            04  FE  A5 E9 001D1           BLBC    DEV_TYPE_ENTRY_STS, 28$                      ; 1090
                            07  FE  A5 E8 001D5           BLBS    DEV_CLS_STATUS, 29$                          ;
                 08         60      16 E1 001D9 28$:      BBC     #22, (R0), 30$                               ; 1092
                            05      65 E9 001DD           BLBC    ENTRY_STATUS, 30$                            ;
                            54      01 90 001E0 29$:      MOVB    #1, INCLUDE_STATUS                           ; 1094
                                    02 11 001E3           BRB     31$                                          ;
                                    54 94 001E5 30$:      CLRB    INCLUDE_STATUS                               ; 1096
                 2F         60      14 E1 001E7 31$:      BBC     #20, (R0), 36$                               ; 1099
                 2B         60      16 E1 001EB           BBC     #22, (R0), 36$                               ; 1100
                                    54 94 001EF           CLRB    INCLUDE_STATUS                               ; 1103
```

```
                        11    FC    A5  E9 001F1          BLBC    DEV_SELECTION_REQUIRED, 33$                 ; 1105
                        04          65  E9 001F5          BLBC    ENTRY_STATUS, 32$                           ; 1108
                        1B    FD    A5  E8 001F8          BLBS    DEVICE_STATUS, 35$
                        1A    FF    A5  E9 001FC 32$:      BLBC    DEV_TYPE_ENTRY_STS, 36$                     ; 1109
                        16    FD    A5  E9 00200          BLBC    DEVICE_STATUS, 36$
                                    11  11 00204          BRB     35$                                         ; 1111
                        51    FF    A5  9A 00206 33$:      MOVZBL  DEV_TYPE_ENTRY_STS, R1                      ; 1115
                        07          51  E9 0020A          BLBC    R1, 34$
                        06    FD    A5  E8 0020D          BLBS    DEVICE_STATUS, 35$
                        06          51  E8 00211          BLBS    R1, 36$                                     ; 1122
                        03          65  E9 00214 34$:      BLBC    ENTRY_STATUS, 36$
                        54          01  90 00217 35$:      MOVB    #1, INCLUDE_STATUS                          ; 1124
                2F      60          15  E1 0021A 36$:      BBC     #21, (R0), 41$                              ; 1129
                2B      60          16  E1 0021E          BBC     #22, (R0), 41$                              ; 1130
                                    54  94 00222          CLRB    INCLUDE_STATUS                              ; 1133
                        11    FC    A5  E9 00224          BLBC    DEV_SELECTION_REQUIRED, 38$                 ; 1135
                        04          65  E9 00228          BLBC    ENTRY_STATUS, 37$                           ; 1138
                        1B    FE    A5  E8 0022B          BLBS    DEV_CLS_STATUS, 40$
                        1A    FF    A5  E9 0022F 37$:      BLBC    DEV_TYPE_ENTRY_STS, 41$                     ; 1139
                        16    FE    A5  E9 00233          BLBC    DEV_CLS_STATUS, 41$
                                    11  11 00237          BRB     40$                                         ; 1141
                        50    FF    A5  9A 00239 38$:      MOVZBL  DEV_TYPE_ENTRY_STS, R0                      ; 1145
                        07          50  E9 0023D          BLBC    R0, 39$
                        06    FE    A5  E8 00240          BLBS    DEV_CLS_STATUS, 40$
                        06          50  E8 00244          BLBS    R0, 41$                                     ; 1152
                        03          65  E9 00247 39$:      BLBC    ENTRY_STATUS, 41$
                        54          01  90 0024A 40$:      MOVB    #1, INCLUDE_STATUS                          ; 1154
                        50          6A  D0 0024D 41$:      MOVL    OPTION_FLAG, R0                             ; 1165
                03      60          04  E0 00250          BBS     #4, (R0), 42$
                                00F4    31 00254          BRW     64$
            00000000G   00          01  D0 00257 42$:      MOVL    #1, EXCLUDE_FLAG                            ; 1168
                        0E    FF    A5  E8 0025E          BLBS    DEV_TYPE_ENTRY_STS, 43$                     ; 1170
                0040    8F          66  B1 00262          CMPW    EMB+4, #64                                  ; 1171
                                    07  13 00267          BEQL    43$
                0041    8F          66  B1 00269          CMPW    EMB+4, #65                                  ; 1172
                                    34  12 0026E          BNEQ    47$
                        50          67  D0 00270 43$:      MOVL    EXCLUDE_MASK, R0                            ; 1175
                13      60          14  E1 00275          BBC     #20, (R0), 45$
            00C00000V   00          00  FB 00277          CALLS   #0, VERIFY_DEVICE                           ; 1178
                        06          50  E9 0027E          BLBC    R0, 44$
                  FD    A5          01  90 00281          MOVB    #1, DEVICE_STATUS                           ; 1180
                                    03  11 00285          BRB     45$
                        FD    A5  94  00287 44$:          CLRB    DEVICE_STATUS                               ; 1182
                        50          67  D0 0028A 45$:      MOVL    EXCLUDE_MASK, R0                            ; 1185
                13      60          15  E1 0028D          BBC     #21, (R0), 47$
            00000000V   00          00  FB 00291          CALLS   #0, VERIFY_DEVICE_CLASS                     ; 1188
                        06          50  E9 00298          BLBC    R0, 46$
                  FE    A5          01  90 0029B          MOVB    #1, DEV_CLS_STATUS                          ; 1190
                                    03  11 0029F          BRB     47$
                        FE    A5  94  002A1 46$:          CLRB    DEV_CLS_STATUS                              ; 1192
                        50          67  D0 002A4 47$:      MOVL    EXCLUDE_MASK, R0                            ; 1196
                11      60          16  E1 002A7          BBC     #22, (R0), 49$
            00000000V   00          00  FB 002AB          CALLS   #0, VERIFY_ENTRY                            ; 1199
                        05          50  E9 002B2          BLBC    R0, 48$
                        65          01  90 002B5          MOVB    #1, ENTRY_STATUS                            ; 1201
                                    02  11 002B8          BRB     49$
                                    65  94 002BA 48$:      CLRB    ENTRY_STATUS                                ; 1203
```

```
                50              67  D0 002BC 49$:   MOVL    EXCLUDE_MASK, R0                        ; 1206
        08      60              14  E1 002BF        BBC     #20, (R0), 50$
                04 FF   A5      E9 002C3        BLBC    DEV_TYPE_ENTRY_STS, 50$                 ; 1207
                13 FD   A5      E8 002C7        BLBS    DEVICE_STATUS, 52$
        08      60              15  E1 002CB 50$:   BBC     #21, (R0), 51$                      ; 1209
                04 FF   A5      E9 002CF        BLBC    DEV_TYPE_ENTRY_STS, 51$                 ; 1210
                07 FE   A5      E8 002D3        BLBS    DEV_CLS_STATUS, 52$
        07      60              16  E1 002D7 51$:   BBC     #22, (R0), 53$                      ; 1212
                04              65  E9 002DB        BLBC    ENTRY_STATUS, 53$
                                53  94 002DE 52$:   CLRB    EXCLUDE_STATUS                      ; 1214
                                03  11 002E0        BRB     54$
                53              01  90 002E2 53$:   MOVB    #1, EXCLUDE_STATUS                  ; 1216
        2F      60              14  E1 002E5 54$:   BBC     #20, (R0), 59$                      ; 1219
        2B      60              16  E1 002E9        BBC     #22, (R0), 59$                      ; 1220
                53              01  90 002ED        MOVB    #1, EXCLUDE_STATUS                  ; 1223
                11 FC   A5      E9 002F0        BLBC    DEV_SELECTION_REQUIRED, 56$             ; 1225
                04              65  E9 002F4        BLBC    ENTRY_STATUS, 55$                   ; 1228
                1B FD   A5      E8 002F7        BLBS    DEVICE_STATUS, 58$
                19 FF   A5      E9 002FB 55$:   BLBC    DEV_TYPE_ENTRY_STS, 59$                 ; 1229
                15 FD   A5      E9 002FF        BLBC    DEVICE_STATUS, 59$
                                11  11 00303        BRB     58$                                 ; 1231
                51 FF   A5      9A 00305 56$:   MOVZBL  DEV_TYPE_ENTRY_STS, R1                  ; 1235
                07              51  E9 00309        BLBC    R1, 57$
                06 FD   A5      E8 0030C        BLBS    DEVICE_STATUS, 58$
                05              51  E8 00310        BLBS    R1, 59$                             ; 1242
                02              65  E9 00313 57$:   BLBC    ENTRY_STATUS, 59$
                                53  94 00316 58$:   CLRB    EXCLUDE_STATUS                      ; 1244
        2F      60              15  E1 00318 59$:   BBC     #21, (R0), 64$                      ; 1249
        2B      60              16  E1 0031C        BBC     #22, (R0), 64$                      ; 1250
                53              01  90 00320        MOVB    #1, EXCLUDE_STATUS                  ; 1253
                11 FC   A5      E9 00323        BLBC    DEV_SELECTION_REQUIRED, 61$             ; 1255
                04              65  E9 00327        BLBC    ENTRY_STATUS, 60$                   ; 1258
                1B FE   A5      E8 0032A        BLBS    DEV_CLS_STATUS, 63$
                19 FF   A5      E9 0032E 60$:   BLBC    DEV_TYPE_ENTRY_STS, 64$                 ; 1259
                15 FE   A5      E9 00332        BLBC    DEV_CLS_STATUS, 64$
                                11  11 00336        BRB     63$                                 ; 1261
                50 FF   A5      9A 00338 61$:   MOVZBL  DEV_TYPE_ENTRY_STS, R0                  ; 1265
                07              50  E9 0033C        BLBC    R0, 62$
                06 FE   A5      E8 0033F        BLBS    DEV_CLS_STATUS, 63$
                05              50  E8 00343        BLBS    R0, 64$                             ; 1272
                02              65  E9 00346 62$:   BLBC    ENTRY_STATUS, 64$
                                53  94 00349 63$:   CLRB    EXCLUDE_STATUS                      ; 1274
                32              54  E9 0034B 64$:   BLBC    INCLUDE_STATUS, 67$                 ; 1288
                2F              53  E9 0034E        BLBC    EXCLUDE_STATUS, 67$
                50              6B  D0 00351        MOVL    PARSER_DATA, R0                     ; 1289
                02              60  91 00354        CMPB    (R0), #2
                                27  12 00357        BNEQ    67$
                50              66  3C 00359        MOVZWL  EMB+4, R0                           ; 1295
      0063      8F              50  B1 0035C        CMPW    R0, #99
                                07  13 00361        BEQL    65$
      0064      8F              50  B1 00363        CMPW    R0, #100                            ; 1296
                                0C  12 00368        BNEQ    66$
                F8 A5           9F 0036A 65$:   PUSHAB  LSTLUN                              ; 1302
00000000G       00              01  FB 0036D        CALLS   #1, INTERVENE_INCREMENT
                                0A  11 00374        BRB     67$
                F8 A5           9F 00376 66$:   PUSHAB  LSTLUN                              ; 1308
00000000G       00              01  FB 00379        CALLS   #1, INTERVENE_OUTPUT
```

```
                        09 00000000G   00  E9 00380 67$:    BLBC    UNKNOWN_ENTRY, 68$                      ; 1316
                        50             67  D0 00387         MOVL    EXCLUDE_MASK, R0                        ; 1321
                     08 60             13  E1 0038A         BBC     #19, (R0), 69$                          ;
                                       0A  11 0038E         BRB     70$                                     ; 1329
                        07             54  E9 00390 68$:    BLBC    INCLUDE_STATUS, 70$                     ; 1333
                        04             53  E9 00393         BLBC    EXCLUDE_STATUS, 70$                     ; 1334
                        50             01  D0 00396 69$:    MOVL    #1, R0                                  ; 1346
                                           04 00399         RET                                             ;
                                       50  D4 0039A 70$:    CLRL    R0                                      ; 1348
                                           04 0039C         RET                                             ;

; Routine Size:  925 bytes,    Routine Base:  $CODE + 0000

;  769          1349  1
;  770          1350  1
```

```
;  772          1351  1 ROUTINE VERIFY_ENTRY =
;  773          1352  2 Begin
;  774          1353  2
;  775          1354  2 !++
;  776          1355  2 !
;  777          1356  2 ! Functional Description:
;  778          1357  2 !
;  779          1358  2 !       This routine will determine if the current entry matches
;  780          1359  2 !       any of the selected entry types. It return TRUE if the
;  781          1360  2 !       current entry matches or return FALSE if the current entry
;  782          1361  2 !       does NOT match.
;  783          1362  2 !
;  784          1363  2 ! Calling sequence:
;  785          1364  2 !
;  786          1365  2 !       VERIFY_ENTRY ()
;  787          1366  2 !
;  788          1367  2 ! Input parameters:
;  789          1368  2 !
;  790          1369  2 !       None
;  791          1370  2 !
;  792          1371  2 ! Output parameters:
;  793          1372  2 !
;  794          1373  2 !       None
;  795          1374  2 !
;  796          1375  2 !--
;  797          1376  2
;  798          1377  2
;  799          1378  2 !
;  800          1379  2 ! Initialize a status indicator.
;  801          1380  2 !
;  802          1381  2 Dev_selection_required = false ;
;  803          1382  2
;  804          1383  2 !
;  805          1384  2 ! Determine if device attention entries are selected.
;  806          1385  2 !
;  807          1386  3 If ((.exclude_mask[exc$v_dev_attentions]) OR
;  808          1387  3     (.include_mask[inc$v_dev_attentions]))
;  809          1388  2 Then
;  810          1389  2     !
;  811          1390  2     ! Determine if this entry is for a device attention.
;  812          1391  2     !
;  813          1392  3     Begin
;  814          1393  3     Dev_selection_required = true ;
;  815          1394  3     If .emb[emb$w_hd_entry] EQLU EMB$K_DA
;  816          1395  3     Then
;  817          1396  3         !
;  818          1397  3         ! Indicate that this entry does match a selected entry
;  819          1398  3         ! type, by returning to the calling routine with a
;  820          1399  3         ! true value.
;  821          1400  3         !
;  822          1401  3         Return true ;
;  823          1402  2     End ;
;  824          1403  2
;  825          1404  2 !
;  826          1405  2 ! Determine if bugcheck entries are selected.
;  827          1406  2 !
;  828          1407  3 If ((.exclude_mask[exc$v_bugchks]) OR
```

```
;  629          1408  3     (.include_mask[inc$v_bugchks]))
;  830          1409  2 Then
;  831          1410  2     !
;  832          1411  2     ! Determine if this entry is for a bugcheck.
;  833          1412  2     !
;  834          1413  3     Begin
;  835          1414  3     Incr I from 0 to 2 do
;  836          1415  4         Begin
;  837          1416  4         If .emb[emb$w_hd_entry] EQLU .bugchks[.I]
;  838          1417  4         Then
;  839          1418  4             !
;  840          1419  4             ! Indicate that this entry does match a selected
;  841          1420  4             ! entry type, by returning to the calling routine
;  842          1421  4             ! with a true value.
;  843          1422  4             !
;  844          1423  4             Return true ;
;  845          1424  3         End ;
;  846          1425  2     End ;
;  847          1426  2
;  848          1427  2 !
;  849          1428  2 ! Determine if 'control entries' are selected.
;  850          1429  2 !
;  851          1430  3 If ((.exclude_mask[exc$v_control_entry]) OR
;  852          1431  3     (.include_mask[inc$v_control_entry]))
;  853          1432  2 Then
;  854          1433  2     !
;  855          1434  2     ! Determine if this entry is a 'control entry'.
;  856          1435  2     !
;  857          1436  3     Begin
;  858          1437  3     Incr I from 0 to 6 do
;  859          1438  4         Begin
;  860          1439  4         If .emb[emb$w_hd_entry] EQLU .control[.I]
;  861          1440  4         Then
;  862          1441  4             !
;  863          1442  4             ! Indicate that this entry does match a selected
;  864          1443  4             ! entry type, by returning to the calling routine
;  865          1444  4             ! with a true value.
;  866          1445  4             !
;  867          1446  4             Return true ;
;  868          1447  3         End ;
;  869          1448  2     End ;
;  870          1449  2
;  871          1450  2 !
;  872          1451  2 ! Determine if 'cpu entries' are selected.
;  873          1452  2 !
;  874          1453  3 If ((.exclude_mask[exc$v_cpu_entry]) OR
;  875          1454  3     (.include_mask[inc$v_cpu_entry]))
;  876          1455  2 Then
;  877          1456  2     !
;  878          1457  2     ! Determine if this entry is a 'cpu entry'.
;  879          1458  2     !
;  880          1459  3     Begin
;  881          1460  3     Incr I from 0 to 7 do
;  882          1461  4         Begin
;  883          1462  4         If .emb[emb$w_hd_entry] EQLU .cpu[.I]
;  884          1463  4         Then
;  885          1464  4             !
```

```
;  886          1465  4              ! Indicate that this entry does match a selected
;  887          1466  4              ! entry type, by returning to the calling routine
;  888          1467  4              ! with a true value.
;  889          1468  4              !
;  890          1469  4              Return true ;
;  891          1470  3          End ;
;  892          1471  2      End ;
;  893          1472  2
;  894          1473  2  !
;  895          1474  2  ! Determine if device errors are selected.
;  896          1475  2  !
;  897          1476  3  If ((.exclude_mask[exc$v_dev_errors]) OR
;  898          1477  3      (.include_mask[inc$v_dev_errors]))
;  899          1478  2  Then
;  900          1479  2      !
;  901          1480  2      ! Determine if this entry is a device error.
;  902          1481  2      !
;  903          1482  3      Begin
;  904          1483  3      Dev_selection_required = true ;
;  905          1484  3
;  906          1485  3      Incr I from 0 to 2 do
;  907          1486  4          Begin
;  908          1487  4          If .emb[emb$w_hd_entry] EQLU .dev_errors[.I]
;  909          1488  4          Then
;  910          1489  4              !
;  911          1490  4              ! Indicate that this entry does match a selected
;  912          1491  4              ! entry type, by returning to the calling routine
;  913          1492  4              ! with a true value.
;  914          1493  4              !
;  915          1494  4              Return true ;
;  916          1495  3          End ;
;  917          1496  2      End ;
;  918          1497  2
;  919          1498  2  !
;  920          1499  2  ! Determine if machine checks are selected.
;  921          1500  2  !
;  922          1501  3  If ((.exclude_mask[exc$v_machine_chks]) OR
;  923          1502  3      (.include_mask[inc$v_machine_chks]))
;  924          1503  2  Then
;  925          1504  2      !
;  926          1505  2      ! Determine if this entry is a machine check.
;  927          1506  2      !
;  928          1507  3      Begin
;  929          1508  3      If .emb[emb$w_hd_entry] EQLU EMB$K_MC
;  930          1509  3      Then
;  931          1510  3          !
;  932          1511  3          ! Indicate that this entry does match a selected
;  933          1512  3          ! entry type, by returning to the calling routine
;  934          1513  3          ! with a true value.
;  935          1514  3          !
;  936          1515  3          Return true ;
;  937          1516  2      End ;
;  938          1517  2
;  939          1518  2  !
;  940          1519  2  ! Determine if memory entries are selected.
;  941          1520  2  !
;  942          1521  3  If ((.exclude_mask[exc$v_memory]) OR
```

```
;  943          1522  3     (.include_mask[inc$v_memory]))
;  944          1523  2 Then
;  945          1524  2     !
;  946          1525  2     ! Determine if this entry is a 'memory entry'.
;  947          1526  2     !
;  948          1527  3     Begin
;  949          1528  3     Incr I from 0 to 1 do
;  950          1529  4         Begin
;  951          1530  4         If .emb[emb$w_hd_entry] EQLU .memorys[.I]
;  952          1531  4         Then
;  953          1532  4             !
;  954          1533  4             ! Indicate that this entry does match a selected
;  955          1534  4             ! entry type, by returning to the calling routine
;  956          1535  4             ! with a true value.
;  957          1536  4             !
;  958          1537  4             Return true ;
;  959          1538  3         End ;
;  960          1539  2     End ;
;  961          1540  2
;  962          1541  2 !
;  963          1542  2 ! Determine if device timeouts are selected.
;  964          1543  2 !
;  965          1544  3 If ((.exclude_mask[exc$v_dev_timeouts]) OR
;  966          1545  3     (.include_mask[inc$v_dev_timeouts]))
;  967          1546  2 Then
;  968          1547  2     !
;  969          1548  2     ! Determine if this entry is a device timeouts.
;  970          1549  2     !
;  971          1550  3     Begin
;  972          1551  3     Dev_selection_required = true ;
;  973          1552  3
;  974          1553  3     If .emb[emb$w_hd_entry] EQLU EMB$K_DT
;  975          1554  3     Then
;  976          1555  3         !
;  977          1556  3         ! Indicate that this entry does match a selected
;  978          1557  3         ! entry type, by returning to the calling routine
;  979          1558  3         ! with a true value.
;  980          1559  3         !
;  981          1560  3         Return true ;
;  982          1561  2     End ;
;  983          1562  2
;  984          1563  2
;  985          1564  2 !
;  986          1565  2 ! Determine if unknown entries have been selected.
;  987          1566  2 ! If unknown entries have not been excluded, then see if this is an
;  988          1567  2 ! unknown entry. If it is set UNKNOWN_ENTRY true.
;  989          1568  2 !
;  990          1569  2 ! Initialize the unknown entry indicator (not an unknown entry).
;  991          1570  2 !
;  992          1571  3 If ((.exclude_mask[exc$v_unknown_entry]) OR
;  993          1572  3     (.include_mask[inc$v_unknown_entry]))
;  994          1573  2 Then
;  995          1574  2     !
;  996          1575  2     ! Determine if this is an unknown entry.
;  997          1576  2     !
;  998          1577  3     Begin
;  999          1578  3     If .unknown_entry
```

```
;  1000          1579  3     Then  Return true ;
;  1001          1580  2     End ;
;  1002          1581  2
;  1003          1582  2 !
;  1004          1583  2 ! Determine if unsolicited mscp entries are selected.
;  1005          1584  2 !
;  1006          1585  3 If ((.exclude_mask[exc$v_unsol_mscp]) OR
;  1007          1586  3     (.include_mask[inc$v_unsol_mscp]))
;  1008          1587  2 Then
;  1009          1588  2     !
;  1010          1589  2     ! Determine if this entry is an unsolicited mscp entry.
;  1011          1590  2     !
;  1012          1591  3     Begin
;  1013          1592  3     If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP
;  1014          1593  3     Then
;  1015          1594  3         !
;  1016          1595  3         ! Indicate that this entry does match a selected
;  1017          1596  3         ! entry type, by returning to the calling routine
;  1018          1597  3         ! with a true value.
;  1019          1598  3         !
;  1020          1599  3         Return true ;
;  1021          1600  2     End ;
;  1022          1601  2
;  1023          1602  2 !
;  1024          1603  2 ! Determine if volume changes are to be excluded.
;  1025          1604  2 !
;  1026          1605  4 If ((.exclude_mask[exc$v_volume])
;  1027          1606  3     OR (.include_mask[inc$v_volume]))
;  1028          1607  2 Then
;  1029          1608  2     !
;  1030          1609  2     ! Determine if this entry is a volume entry.
;  1031          1610  2     !
;  1032          1611  3     Begin
;  1033          1612  3     Dev_selection_required = true ;
;  1034          1613  3
;  1035          1614  3     Incr I from 0 to 1 do
;  1036          1615  4         Begin
;  1037          1616  4         If .emb[emb$w_hd_entry] EQLU .volume[.I]
;  1038          1617  4         Then
;  1039          1618  4             !
;  1040          1619  4             ! Indicate that this entry does match a selected
;  1041          1620  4             ! entry type, by returning to the calling routine
;  1042          1621  4             ! with a true value.
;  1043          1622  4             !
;  1044          1623  4             Return true ;
;  1045          1624  3         End ;
;  1046          1625  2     End ;
;  1047          1626  2
;  1048          1627  2 !
;  1049          1628  2 ! Indicate that this entry does not match any of the selected
;  1050          1629  2 ! entry types, by returning to the calling routine with a
;  1051          1630  2 ! false value.
;  1052          1631  2 !
;  1053          1632  2 Return false ;
;  1054          1633  1 End ;   ! Routine
```

```
                                003C 00000 VERIFY_ENTRY:
                                           .WORD   Save R2,R3,R4,R5                                ; 1351
                55 00000000G   00   9E 00002       MOVAB   EMB+4, R5
                54 00000000'   00   9E 00009       MOVAB   DEV_SELECTION_REQUIRED, R4
                53 00000000G   00   9E 00010       MOVAB   INCLUDE_MASK, R3
                               64   94 00017       CLRB    DEV_SELECTION_REQUIRED                  ; 1381
                51 00000000G   00   D0 00019       MOVL    EXCLUDE_MASK, R1                        ; 1386
          07    61             09   E0 00020       BBS     #9, (R1), 1$
                50             63   D0 00024       MOVL    INCLUDE_MASK, R0                        ; 1387
          0A    60             09   E1 00027       BBC     #9, (R0), 2$
                64             01   90 0002B 1$:    MOVB    #1, DEV_SELECTION_REQUIRED              ; 1393
        0062    8F             65   B1 0002E       CMPW    EMB+4, #98                              ; 1394
                               7D   13 00033       BEQL    16$
          07    61             0A   E0 00035 2$:    BBS     #10, (R1), 3$                           ; 1407
                50             63   D0 00039       MOVL    INCLUDE_MASK, R0                        ; 1408
          10    60             0A   E1 0003C       BBC     #10, (R0), 5$
                               50   D4 00040 3$:    CLRL    I                                       ; 1416
                52       0C A440   9A 00042 4$:    MOVZBL  BUGCHKS[I], R2
                65             52   B1 00047       CMPW    R2, EMB+4
                               7D   13 0004A       BEQL    ^0$
          F2    50             02   F3 0004C       AOBLEQ  #2, I, 4$                               ; 1414
          07    61             0B   E0 00050 5$:    BBS     #11, (R1), 6$                           ; 1430
                50             63   D0 00054       MOVL    INCLUDE_MASK, R0                        ; 1431
          10    60             0B   E1 00057       BBC     #11, (R0), 8$
                               50   D4 0005B 6$:    CLRL    I                                       ; 1439
                52       10 A440   9A 0005D 7$:    MOVZBL  CONTROL[I], R2
                65             52   B1 00062       CMPW    R2, EMB+4
                               7B   13 00065       BEQL    23$
          F2    50             06   F3 00067       AOBLEQ  #6, I, 7$                               ; 1437
          07    61             0C   E0 0006B 8$:    BBS     #12, (R1), 9$                           ; 1453
                50             63   D0 0006F       MOVL    INCLUDE_MASK, R0                        ; 1454
          10    60             0C   E1 00072       BBC     #12, (R0), 11$
                               50   D4 00076 9$:    CLRL    I                                       ; 1462
                52       18 A440   9A 00078 10$:   MOVZBL  CPU[I], R2
                65             52   B1 0007D       CMPW    R2, EMB+4
                               60   13 00080       BEQL    23$
          F2    50             07   F3 00082       AOBLEQ  #7, I, 10$                              ; 1460
          07    61             0D   E0 00086 11$:   BBS     #13, (R1), 12$                          ; 1476
                50             63   D0 0008A       MOVL    INCLUDE_MASK, R0                        ; 1477
          13    60             0D   E1 0008D       BBC     #13, (R0), 14$
                64             01   90 00091 12$:   MOVB    #1, DEV_SELECTION_REQUIRED              ; 1483
                               50   D4 00094       CLRL    I                                       ; 148?
                52       20 A440   9A 00096 13$:   MOVZBL  DEV_ERRORS[I], R2
                65             52   B1 0009B       CMPW    R2, EMB+4
                               66   13 0009E       BEQL    28$
          F2    50             02   F3 000A0       AOBLEQ  #2, I, 13$                              ; 1485
          07    61             0E   E0 000A4 14$:   BBS     #14, (R1), 15$                          ; 1501
                50             63   D0 000A8       MOVL    INCLUDE_MASK, R0                        ; 1502
          05    60             0E   E1 000AB       BBC     #14, (R0), 17$
                02             55   B1 000AF 15$:   CMPW    EMB+4, #2                               ; 1508
                               6E   13 000B2 16$:   BEQL    32$
                               61   B5 000B4 17$:   TSTW    (R1)                                    ; 1521
                               07   19 000B6       BLSS    18$
                50             63   D0 000B8       MOVL    INCLUDE_MASK, R0                        ; 1522
```

```
                                                60  B5 000BB          TSTW    (R0)                                    ;
                                                10  18 000BD          BGEQ    21$                                     ;
                                                50  D4 000BF 18$:     CLRL    I                                       ; 1530
                                        52      24 A440 9A 000C1 19$:     MOVZBL  MEMORYS[I], R2                      ;
                                        65          52  B1 000C6          CMPW    R2, EMB+4                           ;
                                                    57  13 000C9 20$:     BEQL    32$                                 ;
                                    F2      50          01  F3 000CB          AOBLEQ  #1, I, 19$                      ; 1528
                                        07      02      A1  E8 000CF 21$:     BLBS    2(R1), 22$                      ; 1544
                                        50          63  D0 000D3          MOVL    INCLUDE_MASK, R0                    ; 1545
                                        0A      02      A0  E9 000D6          BLBC    2(R0), 24$                      ;
                                        64          01  90 000DA 22$:     MOVB    #1, DEV_SELECTION_REQUIRED          ; 1551
                                0060    8F          65  B1 000DD          CMPW    EMB+4, #96                          ; 1553
                                                    3E  13 000E2 23$:     BEQL    32$                                 ;
                                    07      61          13  E0 000E4 24$:     BBS     #19, (R1), 25$                  ; 1571
                                        50          63  D0 000E8          MOVL    INCLUDE_MASK, R0                    ; 1572
                                    07      60          13  E1 000EB          BBC     #19, (R0), 26$                  ;
                                        2C  00000000G   00  E8 000EF 25$:     BLBS    UNKNOWN_ENTRY, 32$              ; 1578
                                    07      61          11  E0 000F6 26$:     BBS     #17, (R1), 27$                  ; 1585
                                        50          63  D0 000FA          MOVL    INCLUDE_MASK, R0                    ; 1586
                                    07      60          11  E1 000FD          BBC     #17, (R0), 29$                  ;
                                0065    8F          65  B1 00101 27$:     CMPW    EMB+4, #101                         ; 1592
                                                    1A  13 00106 28$:     BEQL    32$                                 ;
                                    07      61          12  E0 00108 29$:     BBS     #18, (R1), 30$                  ; 1605
                                        50          63  D0 0010C          MOVL    INCLUDE_MASK, R0                    ; 1606
                                    17      60          12  E1 0010F          BBC     #18, (R0), 34$                  ;
                                        64          01  90 00113 30$:     MOVB    #1, DEV_SELECTION_REQUIRED          ; 1612
                                                    50  D4 00116          CLRL    I                                   ; 1616
                                        51      28 A440 9A 00118 31$:     MOVZBL  VOLUME[I], R1                       ;
                                        65          51  B1 0011D          CMPW    R1, EMB+4                           ;
                                                    04  12 00120          BNEQ    33$                                 ;
                                        50          01  D0 00122 32$:     MOVL    #1, R0                              ; 1623
                                                    04  00125          RET                                         ;
                                    EE      50          01  F3 00126 33$:     AOBLEQ  #1, I, 31$                      ; 1614
                                                    50  D4 0012A 34$:     CLRL    R0                                  ; 1632
                                                        04  0012C          RET                                         ; 1633

; Routine Size:  301 bytes,    Routine Base:  $CODE + 039D

; 1055          1634  1
```

```
;1057     1635  1 GLOBAL ROUTINE DEVICE_TYPE_ENTRY =
;1058     1636  2 Begin
;1059     1637  2
;1060     1638  2 !++
;1061     1639  2 !
;1062     1640  2 ! Functional Description:
;1063     1641  2 !
;1064     1642  2 !       This routine will determine if the current entry is a device
;1065     1643  2 !       type entry; (device attention, device error, device timeout,
;1066     1644  2 !       volume dismount, volume mount). It return TRUE if the current
;1067     1645  2 !       entry matches any of the device type entries or return FALSE
;1068     1646  2 !       if the current entry does NOT match.
;1069     1647  2 !
;1070     1648  2 ! Calling sequence:
;1071     1649  2 !
;1072     1650  2 !       DEVICE_ENTRY_TYPE ()
;1073     1651  2 !
;1074     1652  2 ! Input parameters:
;1075     1653  2 !
;1076     1654  2 !       None
;1077     1655  2 !
;1078     1656  2 ! Output parameters:
;1079     1657  2 !
;1080     1658  2 !       None
;1081     1659  2 !
;1082     1660  2 !--
;1083     1661  2
;1084     1662  2 OWN
;1085     1663  2     Device_entries:     VECTOR [6,byte,unsigned] ! Storage for device type
;1086     1664  2                                                  ! entries.
;1087     1665  2                         Initial (BYTE
;1088     1666  2                                 (EMB$K_DA,       ! Device attentions
;1089     1667  2                                 EMB$K_DE,        ! Device errors
;1090     1668  2                                 EMB$K_DT,        ! Device timeouts
;1091     1669  2                                 EMB$K_LM,
;1092     1670  2                                 EMB$K_SP,        ! Log message
;1093     1671  2                                 EMB$K_LOGMSCP)   ;! Unsolicited mscp msg
;1094     1672  2
;1095     1673  2 !
;1096     1674  2 ! Determine if the current entry is a device type entry.
;1097     1675  2 !
;1098     1676  2 Incr I from 0 to 5 do
;1099     1677  3     Begin
;1100     1678  3     If .emb[emb$w_hd_entry] EQLU .device_entries[.I]
;1101     1679  3     Then
;1102     1680  3         !
;1103     1681  3         ! Indicate that this is a device type entry, by
;1104     1682  3         ! returning to the calling routine with a true value.
;1105     1683  3         !
;1106     1684  3         Return true ;
;1107     1685  2     End ;
;1108     1686  2
;1109     1687  2 !
;1110     1688  2 ! Indicate that this is NOT a device type entry, by returning
;1111     1689  2 ! to the calling routine with a false value.
;1112     1690  2 !
;1113     1691  2 Return false ;
```

```
;  1114          1692  2
;  1115          1693  1 End ;    ! Routine


                                                        .PSECT  $OWN$,NOEXE,  PIC,2

                                                 0002E  .BLKB   2
                                 65  63  64  60  01  62  00030 DEVICE_ENTRIES:
                                                        .BYTE   98, 1, 96, 100, 99, 101                 ;


                                                        .PSECT  $CODE,NOWRT,  PIC,2

                                                   0000 00000   .ENTRY  DEVICE_TYPE_ENTRY, Save nothing    ; 1635
                                              50    D4 00002   CLRL   I                                    ; 1678
                                 51 00000000'0040   9A 00004 1$: MOVZBL DEVICE_ENTRIES[I], R1
                      00000000G  00              51  B1 00007   CMPW   R1, EMB+4
                                                 04  12 0000C   BNEQ   2$
                                 50              01  D0 0000E   MOVL   #1, R0                            ; 1684
                                                    04 00011   RET
                          E7     50              05  F3 00012 2$: AOBLEQ #5, I, 1$                      ; 1676
                                              50    D4 00016   CLRL   R0                                   ; 1691
                                                    04 00018   RET                                       ; 1693

;  Routine Size:  25 bytes,    Routine Base:  $CODE + 04CA

;  1116          1694  1
```

```
;  1118          1695  1 ROUTINE VERIFY_DEVICE_CLASS =
;  1119          1696  2 Begin
;  1120          1697  2
;  1121          1698  2 !++
;  1122          1699  2 !
;  1123          1700  2 ! Functional Description:
;  1124          1701  2 !
;  1125          1702  2 !       This routine will determine if the device recorded by the
;  1126          1703  2 !       current entry matches any of the selected device class(es).
;  1127          1704  2 !       It return TRUE if the current entry matches or return FALSE
;  1128          1705  2 !       if the current entry does NOT match.
;  1129          1706  2 !
;  1130          1707  2 ! Calling sequence:
;  1131          1708  2 !
;  1132          1709  2 !       VERIFY_DEVICE_CLASS ()
;  1133          1710  2 !
;  1134          1711  2 ! Input parameters:
;  1135          1712  2 !
;  1136          1713  2 !       None
;  1137          1714  2
;  1138          1715  2 ! Output parameters:
;  1139          1716  2 !
;  1140          1717  2 !       None
;  1141          1718  2 !
;  1142          1719  2 !--
;  1143          1720  2
;  1144          1721  2 !
;  1145          1722  2 ! Determine whether this is a unsolicited mscp entry and
;  1146          1723  2 ! whether to continue.
;  1147          1724  2 !
;  1148          1725  2 If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
;  1149          1726  2     NOT .include_mask[inc$v_disks] AND
;  1150          1727  2      NOT .include_mask[inc$v_tapes]
;  1151          1728  2 Then
;  1152          1729  2     Return false ;
;  1153          1730  2
;  1154          1731  2 !
;  1155          1732  2 ! Determine if 'BUS' entries are selected.
;  1156          1733  2 !
;  1157          1734  3 If ((.exclude_mask[exc$v_buses]) OR
;  1158          1735  3     (.include_mask[inc$v_buses]))
;  1159          1736  2 Then
;  1160          1737  2     !
;  1161          1738  2     ! Determine if the device recorded by this entry, matches the
;  1162          1739  2     ! selected device class.
;  1163          1740  2     !
;  1164          1741  3     Begin
;  1165          1742  5     If  ((.emb[emb$w_hd_entry] EQLU EMB$K_LM AND
;  1166          1743  3         .emb[emb$b_lm_class] EQLU DC$_BUS)) OR
;  1167          1744  3
;  1168          1745  5         ((.emb[emb$w_hd_entry] EQLU EMB$K_SP AND
;  1169          1746  3         .emb[emb$b_sp_class] EQLU DC$_BUS)) OR
;  1170          1747  3
;  1171          1748  4         (.emb_emb$b_dv_class] EQLU DC$_BUS)
;  1172          1749  3     Then
;  1173          1750  3         !
;  1174          1751  3         ! Indicate that this entry does match a selected device
```

```
; 1175     1752  3          ! class, by returning to the calling routine with a
; 1176     1753  3          ! true value.
; 1177     1754  3          !
; 1178     1755  3          Return true ;
; 1179     1756  2      End ;
; 1180     1757  2
; 1181     1758  2  !
; 1182     1759  2  ! Determine if 'DISK' entries are selected.
; 1183     1760  2  !
; 1184     1761  3  If ((.exclude_mask[exc$v_disks]) OR
; 1185     1762  3      (.include_mask[inc$v_disks]))
; 1186     1763  2  Then
; 1187     1764  2          !
; 1188     1765  2          ! Determine if the device recorded by this entry, matches the
; 1189     1766  2          ! selected device class.
; 1190     1767  2          !
; 1191     1768  3          Begin
; 1192     1769  4          If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
; 1193     177'  4              (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
; 1194     177?  3          Then
; 1195     1772  3              !
; 1196     1773  3              ! Determine if the device recorded by this volume
; 1197     1774  3              ! mount or dismount is a 'disk' type device.
; 1198     1775  3              !
; 1199     1776  4              Begin
; 1200     1777  4              If NOT TRANSLATE_CLASS (emb[emb$t_vm_namtxt],DC$_DISK)
; 1201     1778  4              Then
; 1202     1779  4                  !
; 1203     1780  4                  ! Indicate that the device recorded by this entry is
; 1204     1781  4                  ! not a 'disk', by returning to the calling routine
; 1205     1782  4                  ! with a false value.
; 1206     1783  4                  !
; 1207     1784  4                  Return false
; 1208     1785  4              Else
; 1209     1786  4                  Return true ;
; 1210     1787  3              End ;
; 1211     1788  3
; 1212     1789  5      If ( ((.emb[emb$w_hd_entry] EQLU EMB$K_LM) AND
; 1213     1790  4              (.emb[emb$b_lm_class] EQLU DC$_DISK)) OR
; 1214     1791  4
; 1215     1792  5          ((.emb[emb$w_hd_entry] EQLU EMB$K_SP) AND
; 1216     1793  4              (.emb[emb$b_sp_class] EQLU DC$_DISK)) OR
; 1217     1794  4
; 1218     1795  4      ! Entry type must be either a device error, timeout, or attention.
; 1219     1796  4      !
; 1220     1797  4          (.emb[emb$b_dv_class] EQLU DC$_DISK) )
; 122?     1798  3      Then
; .662     1799  3          !
; 1223     1800  3          ! Indicate that this entry does match a selected
; 1224     1801  3          ! device class, by returning to the calling routine
; 1225     1802  3          ! with a true value.
; 1226     1803  3          !
; 1227     1804  3          Return true ;
; 1228     1805  3
; 1229     1806  3      !
; 1230     1807  3      ! Determine whether this is disk related unsolicited mscp entry.
; 1231     1808  3      !
```

```
;  1232     1809  3     If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
;  1233     1810  3         CH$EQL (2,emb[driver_type],2,CH$PTR(uplit('DISK')))
;  1234     1811  3     Then
;  1235     1812  3         ! Yes, return to the calling routine with a true value.
;  1236     1813  3         !
;  1237     1814  3         Return true ;
;  1238     1815  2     End ;
;  1239     1816  2
;  1240     1817  2 !
;  1241     1818  2 ! Determine if 'REALTIME' entries are selected.
;  1242     1819  2 !
;  1243     1820  3 If ((.exclude_mask[exc$v_realtime]) OR
;  1244     1821  3     (.include_mask[inc$v_realtime]))
;  1245     1822  2 Then
;  1246     1823  2     !
;  1247     1824  2     ! Determine if the device recorded by this entry, matches the
;  1248     1825  2     ! selected device class.
;  1249     1826  2     !
;  1250     1827  3     Begin
;  1251     1828  3     If .emb[emb$b_dv_class] EQLU DC$_REALTIME
;  1252     1829  3     Then
;  1253     1830  3         !
;  1254     1831  3         ! Indicate that this entry does match a selected
;  1255     1832  3         ! device class, by returning to the calling routine
;  1256     1833  3         ! with a true value.
;  1257     1834  3         !
;  1258     1835  3         Return true ;
;  1259     1836  2     End ;
;  1260     1837  2
;  1261     1838  2 !
;  1262     1839  2 ! Determine if 'SYNCHRONOUS COMMUNICATION' entries are selected.
;  1263     1840  2 !
;  1264     1841  3 If ((.exclude_mask[exc$v_sync_comm]) OR
;  1265     1842  3     (.include_mask[inc$v_sync_comm]))
;  1266     1843  2 Then
;  1267     1844  2     !
;  1268     1845  2     ! Determine if the device recorded by this entry. matches the
;  1269     1846  2     ! selected device class.
;  1270     1847  2     !
;  1271     1848  3     Begin
;  1272     1849  3     If .emb[emb$b_dv_class] EQLU DC$_SCOM
;  1273     1850  3     Then
;  1274     1851  3         !
;  1275     1852  3         ! Indicate that this entry does match a selected
;  1276     1853  3         ! device class, by returning to the calling routine
;  1277     1854  3         ! with a true value.
;  1278     1855  3         !
;  1279     1856  3         Return true ;
;  1280     1857  2     End ;
;  1281     1858  2
;  1282     1859  2 !
;  1283     1860  2 ! Determine if 'TAPE' entries are selected.
;  1284     1861  2 !
;  1285     1862  3 If ((.exclude_mask[exc$v_tapes]) OR
;  1286     1863  3     (.include_mask[inc$v_tapes]))
;  1287     1864  2 Then
;  1288     1865  2     !
```

```
;  1289     1866  2      ! Determine if the device recorded by this entry, matches the
;  1290     1867  2      ! selected device class.
;  1291     1868  2      !
;  1292     1869  3      Begin
;  1293     1870  4      If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  1294     1871  4          (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
;  1295     1872  3      Then
;  1296     1873  3          !
;  1297     1874  3          ! Determine if the device recorded by this volume
;  1298     1875  3          ! mount or dismount is a 'tape' type device.
;  1299     1876  3          !
;  1300     1877  4          Begin
;  1301     1878  4          If NOT TRANSLATE_CLASS (emb[emb$t_vm_namtxt],DC$_TAPE)
;  1302     1879  4          Then
;  1303     1880  4              !
;  1304     1881  4              ! Indicate that the device recorded by this entry is
;  1305     1882  4              ! not a 'tape', by returning to the calling routine
;  1306     1883  4              ! with a false value.
;  1307     1884  4              !
;  1308     1885  4              Return false
;  1309     1886  4          Else
;  1310     1887  4              Return true ;
;  1311     1888  3          End ;
;  1312     1889  3
;  1313     1890  5      If ( ((.emb[emb$w_hd_entry] EQLU EMB$K_LM) AND
;  1314     1891  4              (.emb[emb$b_lm_class] EQLU DC$_TAPE)) OR
;  1315     1892  4
;  1316     1893  5          ((.emb[emb$w_hd_entry] EQLU EMB$K_SP) AND
;  1317     1894  4              (.emb[emb$b_sp_class] EQLU DC$_TAPE)) OR
;  1318     1895  4
;  1319     1896  4      ! Entry type must be either a device error, timeout, or attention.
;  1320     1897  4      !
;  1321     1898  4          (.emb[emb$b_dv_class] EQLU DC$_TAPE) )
;  1322     1899  3      Then
;  1323     1900  3          !
;  1324     1901  3          ! Indicate that this entry does match a selected
;  1325     1902  3          ! device class, by returning to the calling routine
;  1326     1903  3          ! with a true value.
;  1327     1904  3          !
;  1328     1905  3          Return true ;
;  1329     1906  3
;  1330     1907  2      !
;  1331     1908  2      ! Determine whether this is tape related unsolicited mscp entry.
;  1332     1909  2      !
;  1333     1910  2      If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
;  1334     1911  2          CH$EQL (2,emb[driver_type],2,CH$PTR(uplit('TAPE')))
;  1335     1912  2      Then
;  1336     1913  2          ! Yes, return to the calling routine with a true value.
;  1337     1914  2          !
;  1338     1915  2          Return true ;
;  1339     1916  2      End ;
;  1340     1917  2
;  1341     1918  2 !
;  1342     1919  2 ! Determine if 'MISC' entries are selected.
;  1343     1920  2 !
;  1344     1921  2 !If ((.exclude_mask[exc$v_misc]) OR
;  1345     1922  2 !    (.include_mask[inc$v_misc]))
```

```
;  1346     1923  2 !Then
;  1347     1924  2     !
;  1348     1925  2     ! Determine if the device recorded by this entry, matches the
;  1349     1926  2     ! selected device class.
;  1350     1927  2     !
;  1351     1928  2 !   Begin
;  1352     1929  2 !   If .emb[emb$b_dv_class] EQLU DC$_MISC
;  1353     1930  2 !   Then
;  1354     1931  2         !
;  1355     1932  2         ! Indicate that this entry does match a selected
;  1356     1933  2         ! device class, by returning to the calling routine
;  1357     1934  2         ! with a true value.
;  1358     1935  2         !
;  1359     1936  2 !       Return true ;
;  1360     1937  2 !   End ;
;  1361     1938  2
;  1362     1939  2 !
;  1363     1940  2 ! Determine if 'LP' entries are selected.
;  1364     1941  2 !
;  1365     1942  2 !If ((.exclude_mask[exc$v_line_printr]) OR
;  1366     1943  2 !     (.include_mask[inc$v_line_printr]))
;  1367     1944  2 !Then
;  1368     1945  2     !
;  1369     1946  2     ! Determine if the device recorded by this entry, matches the
;  1370     1947  2     ! selected device class.
;  1371     1948  2     !
;  1372     1949  2 !   Begin
;  1373     1950  2 !   If .emb[emb$b_dv_class] EQLU DC$_LP
;  1374     1951  2 !   Then
;  1375     1952  2         !
;  1376     1953  2         ! Indicate that this entry does match a selected
;  1377     1954  2         ! device class, by returning to the calling routine
;  1378     1955  2         ! with a true value.
;  1379     1956  2         !
;  1380     1957  2 !       Return true ;
;  1381     1958  2 !   End ;
;  1382     1959  2
;  1383     1960  2 !
;  1384     1961  2 ! Determine if 'JOURNAL' entries are selected.
;  1385     1962  2 !
;  1386     1963  2 !If ((.exclude_mask[exc$v_journal]) OR
;  1387     1964  2 !     (.include_mask[inc$v_journal]))
;  1388     1965  2 !Then
;  1389     1966  2     !
;  1390     1967  2     ! Determine if the device recorded by this entry, matches the
;  1391     1968  2     ! selected device class.
;  1392     1969  2     !
;  1393     1970  2 !   Begin
;  1394     1971  2 !   If .emb[emb$b_dv_class] EQLU DC$_JOURNAL
;  1395     1972  2 !   Then
;  1396     1973  2         !
;  1397     1974  2         ! Indicate that this entry does match a selected
;  1398     1975  2         ! device class, by returning to the calling routine
;  1399     1976  2         ! with a true value.
;  1400     1977  2         !
;  1401     1978  2 !       Return true ;
;  1402     1979  2 !   End ;
```

```
;  1403          1980  2
;  1404          1981  2 !
;  1405          1982  2 ! Indicate that this entry does not match any of the selected
;  1406          1983  2 ! device classes, by returning to the calling routine with a
;  1407          1984  2 ! false value.
;  1408          1985  2 !
;  1409          1986  2 Return false ;
;  1410          1987  1 End ;    . Routine


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

                                    4B  53  49  44  00000 P.AAA:  .ASCII  \DISK\                                    ;
                                    45  50  41  54  00004 P.AAB:  .ASCII  \TAPE\                                    ;


                                                        .PSECT  $CODE,NOWRT,  PIC,2

                                                003C 00000 VERIFY_DEVICE_CLASS:
                                                        .WORD   Save R2,R3,R4,R5                                ; 1695
                                    55 00000000G   00   9E 00002        MOVAB   EXCLUDE_MASK, R5                    ;
                                    54 00000000G   00   9E 00009        MOVAB   INCLUDE_MASK, R4                    ;
                                    53 00000C00G   00   9E 00010        MOVAB   EMB+16, R3                          ;
                                    52         F4   A3   3C 00017        MOVZWL  EMB+4, R2                           ; 1725
                              0065  8F             52   B1 0001B        CMPW    R2, #101                            ;
                                                   11   12 00020        BNEQ    1$                                  ;
                                    50             64   D0 00022        MOVL    INCLUDE_MASK, R0                    ; 1726
                              0A    60             02   E0 00025        BBS     #2, (R0), 1$                        ;
                                    50             64   D0 00029        MOVL    INCLUDE_MASK, R0                    ; 1727
                                    03         01  A0   E8 0002C        BLBS    1(R0), 1$                           ;
                                                 010A   31 00030        BRW     28$                                 ;
                                    51             65   D0 00033  1$:   MOVL    EXCLUDE_MASK, R1                    ; 1734
                              07    61             01   E0 00036        BBS     #1, (R1), 2$                        ;
                                    50             64   D0 0003A        MOVL    INCLUDE_MASK, R0                    ; 1735
                              21    60             01   E1 0003D        BBC     #1, (R0), 5$                        ;
                                 0064 8F             52   B1 00041  2$:   CMPW    R2, #100                            ; 1742
                                                   06   12 00046        BNEQ    3$                                  ;
                                   80 8F             63   91 00048        CMPB    EMB+16, #128                        ; 1743
                                                   77   13 0004C        BEQL    13$                                 ;
                                 0063 8F             52   B1 0004E  3$:   CMPW    R2, #99                             ; 1745
                                                   06   12 00053        BNEQ    4$                                  ;
                                   80 8F             63   91 00055        CMPB    EMB+16, #128                        ; 1746
                                                   7B   13 00059        BEQL    16$                                 ;
                                   80 8F         0C  A3   91 0005B  4$:   CMPB    EMB+28, #128                        ; 1748
                                                   74   13 00060        BEQL    16$                                 ;
                              07    61             02   E0 00062  5$:   BBS     #2, (R1), 6$                        ; 1761
                                    50             64   D0 00066        MOVL    INCLUDE_MASK, R0                    ; 1762
                              45    60             02   E1 00069        BBC     #2, (R0), 11$                       ;
                                 0040 8F             52   B1 0006D  6$:   CMPW    R2, #64                             ; 1769
                                                   07   13 00072        BEQL    7$                                  ;
                                 0041 8F             52   B1 00074        CMPW    R2, #65                             ; 1770
                                                   04   12 00079        BNEQ    8$                                  ;
                                                   01   DD 0007B  7$:   PUSHL   #1                                 ; 1777
                                                   78   11 0007D        BRB     20$                                 ;
                                    50         F4   A3   3C 0007F  8$:   MOVZWL  EMB+4, R0                           ; 1789
```

```
                    0064    8F              50  B1 00083          CMPW    R0, #100
                                            05  12 00088          BNEQ    9$
                            01              63  91 0008A          CMPB    EMB+16, #1                      ;1790
                                            47  13 0008D          BEQL    16$
                    0063    8F              50  B1 0008F  9$:     CMPW    R0, #99                         ;1792
                                            05  12 00094          BNEQ    10$
                            01              63  91 00096          CMPB    EMB+16, #1                      ;1793
                                            79  13 00099          BEQL    22$
                            01      0(      A3  91 0009B  10$:     CMPB    EMB+28, #1                      ;1797
                                            7F  13 0009F          BEQL    24$
                    0065    8F              50  B1 000A1          CMPW    R0, #101                        ;1809
                                            0A  12 000A6          BNEQ    11$
                00000000'   00      02      A3  B1 000A8          CMPW    EMB+18, P.AAA                   ;1810
                                            74  13 000B0          BEQL    26$
                            51              65  D0 000B2  11$:     MOVL    EXCLUDE_MASK, R1                ;1820
              07            61              06  E0 000B5          BBS     #6, (R1), 12$
                            50              64  D0 000B9          MOVL    INCLUDE_MASK, R0                ;1821
              07            60              06  E1 000BC          BBC     #6, (R0), 14$
                    60      8F      0C      A3  91 000C0  12$:     CMPB    EMB+28, #96                     ;1828
                                            72  13 000C5  13$:     BEQL    27$
                                            61  95 000C7  14$:     TSTB    (R1)                            ;1841
                                            07  19 000C9          BLSS    15$
                            50              64  D0 000CB          MOVL    INCLUDE_MASK, R0                ;1842
                                            60  95 000CE          TSTB    (R0)
                                            06  18 000D0          BGEQ    17$
                            20      0C      A3  91 000D2  15$:     CMPB    EMB+28, #32                     ;1849
                                            61  13 000D6  16$:     BEQL    27$
                            07      01      A1  E8 000D8  17$:     BLBS    1(R1), 18$                      ;1862
                            50              64  D0 000DC          MOVL    INCLUDE_MASK, R0                ;1863
                            5A      01      A0  E9 000DF          BLBC    1(R0), 28$
                            50      F4      A3  3C 000E3  18$:     MOVZWL  EMB+4, R0                       ;1870
                    0040    8F              50  B1 000E7          CMPW    R0, #64
                                            07  13 000EC          BEQL    19$
                    0041    8F              50  B1 000EE          CMPW    R0, #65                         ;1871
                                            11  12 000F3          BNEQ    21$
                                            02  DD 000F5  19$:     PUSHL   #2                              ;1878
                                    0F      A3  9F 000F7  20$:     PUSHAB  EMB+3,
                00000000V   00              02  FB 000FA          CALLS   #2, TRANSLATE_CLASS
                            35              50  E8 00101          BLBS    R0, 27$
                                            57  11 00104          BRB     28$                             ;1887
                            50      F4      A3  3C 00106  21$:     MOVZWL  EMB+4, R0                       ;1890
                    0064    8F              50  B1 0010A          CMPW    R0, #100
                                            05  12 0010F          BNEQ    23$
                            02              63  91 00111          CMPB    EMB+16, #2                      ;1891
                                            23  13 00114  22$:     BEQL    27$
                    0063    8F              50  B1 00116  23$:     CMPW    R0, #99                         ;1893
                                            05  12 0011B          BNEQ    25$
                            02              63  91 0011D          CMPB    EMB+16, #2                      ;1894
                                            17  13 00120  24$:     BEQL    27$
                            02      0C      A3  91 00122  25$:     CMPB    EMB+28, #2                      ;1898
                                            11  13 00126  26$:     BEQL    27$
                    0065    8F              50  B1 00128          CMPW    R0, #101                        ;1910
                                            0E  12 0012D          BNEQ    28$
                00000000'   00      02      A3  B1 0012F          CMPW    EMB+18, P.AAB                   ;1911
                                            04  12 00137          BNEQ    28$
                            50              01  D0 00139  27$:     MOVL    #1, R0                          ;1915
                                                04 0013C          RET
```

```
                50  D4 0013D 28$:    CLRL    R0                              ; 1987
                    04 0013F         RET                                     ;

; Routine Size:  320 bytes,    Routine Base:  $CODE + 04EA

; 1411          1988  1
```

```
;  1413          1989  1 Routine VERIFY_DEVICE =
;  1414          1990  2 Begin
;  1415          1991  2
;  1416          1992  2 !++
;  1417          1993  2 !
;  1418          1994  2 !--
;  1419          1995  2
;  1420          1996  2 Local
;  1421          1997  2     Dev_name,
;  1422          1998  2     Dev_name_length,
;  1423          1999  2     Dev_unit,
;  1424          2000  2     Status ;
;  1425          2001  2
;  1426          2002  2 Bind
;  1427          2003  2     lm_name_length = emb[emb$t_lm_devnam] : BYTE,
;  1428          2004  2     sp_name_length = emb[emb$t_sp_devnam] : BYTE,
;  1429          2005  2     dv_name_length = emb[emb$t_dv_name] : BYTE ;
;  1430          2006  2
;  1431          2007  2 !
;  1432          2008  2 ! Determine whether this is an unsolicited mscp entry and
;  1433          2009  2 ! return with a false value if so (logmscp entries are not
;  1434          2010  2 ! applicable to a specific device).
;  1435          2011  2 !
;  1436          2012  2 If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP
;  1437          2013  2 Then
;  1438          2014  2     Return false ;
;  1439          2015  2
;  1440          2016  2 !
;  1441          2017  2 ! Determine the type of entry so that the comparison for the
;  1442          2018  2 ! device class is made against the appropriate field in the entry.
;  1443          2019  2 !
;  1444          2020  2 ! Determine if this a log message entry.
;  1445          2021  2 !
;  1446          2022  2 If .emb[emb$w_hd_entry] EQLU EMB$K_LM
;  1447          2023  2 Then
;  1448          2024  2     !
;  1449          2025  2     ! Entry type is a log message, get the device name,
;  1450          2026  2     ! name length, and unit number.
;  1451          2027  2     !
;  1452          2028  3     Begin
;  1453          2029  3     Dev_name = emb[emb$t_lm_devnam] + 1 ;
;  1454          2030  3     Dev_name_length = .lm_name_length ;
;  1455          2031  3     Dev_unit = .emb[emb$w_lm_unit] ;
;  1456          2032  3     End
;  1457          2033  2 Else
;  1458          2034  2     !
;  1459          2035  2     ! Determine if this is a log status entry.
;  1460          2036  2     !
;  1461          2037  3     Begin
;  1462          2038  3     If .emb[emb$w_hd_entry] EQLU EMB$K_SP
;  1463          2039  3     Then
;  1464          2040  3         !
;  1465          2041  3         ! Entry type is a log status, get the device name,
;  1466          2042  3         ! name length, and unit number.
;  1467          2043  3         !
;  1468          2044  4         Begin
;  1469          2045  4         Dev_name = emb[emb$t_sp_devnam] + ' ;
```

```
;  1470          2046  4          Dev_name_length = .sp_name_length ;
;  1471          2047  4          Dev_unit = .emb[emb$w_sp_unit] :
;  1472          2048  4          End
;  1473          2049  3      Else
;  1474          2050  3          !
;  1475          2051  3          ! Determine if this a volume mount/dismount entry.
;  1476          2052  3          !
;  1477          2053  4          Begin
;  1478          2054  5          If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  1479          2055  5              (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
;  1480          2056  4          Then
;  1481          2057  4              !
;  1482          2058  4              ! Entry type is a either a volume mount/dismount, get
;  1483          2059  4              ! the device name, name length, and unit number.
;  1484          2060  4              !
;  1485          2061  5              Begin
;  1486          2062  5              Dev_name = emb[emb$t_vm_namtxt] ;
;  1487          2063  5              Dev_name_length = .emb[emb$b_vm_namlng] ;
;  1488          2064  5              Dev_unit = .emb[emb$w_vm_unit] ;
;  1489          2065  5              End
;  1490          2066  4          Else
;  1491          2067  4              !
;  1492          2068  4              ! Entry type must be either a device error, device timeount,
;  1493          2069  4              ! or a device attention, get the device name, name length, and
;  1494          2070  4              ! unit number.
;  1495          2071  4              !
;  1496          2072  5              Begin
;  1497          2073  5              Dev_name = emb[emb$t_dv_name] + 1 ;
;  1498          2074  5              Dev_name_length = .dv_name_length ;
;  1499          2075  5              Dev_unit = .emb[emb$w_dv_unit] ;
;  1500          2076  4              End ;
;  1501          2077  3          End ;
;  1502          2078  2      End ;
;  1503          2079  2
;  1504          2080  2 !
;  1505          2081  2 ! Call the search queue routine to determine if the device recorded by
;  1506          2082  2 ! this entry matches any of the selected devices.
;  1507          2083  2 !
;  1508          2084  2 Status = SEARCH_QUEUE (.dev_name,dev_name_length,dev_unit) ;
;  1509          2085  2
;  1510          2086  2 !
;  1511          2087  2 ! Return the status from the search queue operation to the
;  1512          2088  2 ! calling routine.
;  1513          2089  2 !
;  1514          2090  2 .Status
;  1515          2091  1 End ;   ! Routine


                                0004 00000 VERIFY_DEVICE:
                                                .WORD   Save R2                 ;  1989
                    52 00000000G   00  9E 00002         MOVAB   EMB+4, R2       ;
                    5E             08  C2 00009         SUBL2   #8, SP          ;
                    50             62  3C 0000C         MOVZWL  EMB+4, R0       ;  2012
              0065  8F             50  B1 0000F         CMPW    R0, #101        ;
```

```
                                        03  12 00014          BNEQ    1$                                                      ;
                                        50  D4 00016          CLRL    R0                                                      ; 2014
                                        04  00018            RET                                                             ;
                0064    8F              50  B1 00019 1$:      CMPW    R0, #100                                                ; 2022
                                        0F  12 0001E          BNEQ    2$                                                      ;
                        51      11      A2  9E 00020          MOVAB   EMB+21, DEV_NAME                                        ; 2029
                  04    AE      10      A2  9A 00024          MOVZBL  LM_NAME_LENGTH, DEV_NAME_LENGTH                         ; 2030
                        6E      0E      A2  3C 00029          MOVZWL  EMB+18, DEV_UNIT                                        ; 2031
                                        3C  11 0002D          BRB     7$                                                      ; 2022
                0063    8F              50  B1 0002F 2$:      CMPW    R0, #99                                                 ; 2038
                                        0B  12 00034          BNEQ    3$                                                      ;
                        51      3D      A2  9E 00036          MOVAB   EMB+65, DEV_NAME                                        ; 2045
                  04    AE      3C      A2  9A 0003A          MOVZBL  SP_NAME_LENGTH, DEV_NAME_LENGTH                         ; 2046
                                        26  11 0003F          BRB     6$                                                      ; 2047
                0040    8F              50  B1 00041 3$:      CMPW    R0, #64                                                 ; 2054
                                        07  13 00046          BEQL    4$                                                      ;
                0041    8F              50  B1 00048          CMPW    R0, #65                                                 ; 2055
                                        0F  12 0004D          BNEQ    5$                                                      ;
                        51      1B      A2  9E 0004F 4$:      MOVAB   EMB+31, DEV_NAME                                        ; 2062
                  04    AE      1A      A2  9A 00053          MOVZBL  EMB+30, DEV_NAME_LENGTH                                 ; 2063
                        6E      18      A2  3C 00058          MOVZWL  EMB+28, DEV_UNIT                                        ; 2064
                                        0D  11 0005C          BRB     7$                                                      ; 2054
                        51      3B      A2  9E 0005E 5$:      MOVAB   EMB+63, DEV_NAME                                        ; 2073
                  04    AE      3A      A2  9A 00062          MOVZBL  DV_NAME_LENGTH, DEV_NAME_LENGTH                         ; 2074
                        6E      26      A2  3C 00067 6$:      MOVZWL  EMB+42, DEV_UNIT                                        ; 2075
                                        5E  DD 0006B 7$:      PUSHL   SP                                                      ; 2084
                                08      AE  9F 0006D          PUSHAB  DEV_NAME_LENGTH                                         ;
                                        51  DD 00070          PUSHL   DEV_NAME                                                ;
          00000000G     00              03  FB 00072          CALLS   #3, SEARCH_QUEUE                                        ;
                                        04  00079            RET                                                             ; 2091

; Routine Size:  122 bytes,    Routine Base:  $CODE + 062A

; 1516          2092  1
```

```
;  1518          2093  1 GLOBAL ROUTINE TRANSLATE_CLASS (search_name,dev_class) =
;  1519          2094  2 Begin
;  1520          2095  2
;  1521          2096  2 !++
;  1522          2097  2 !
;  1523          2098  2 ! Functional Description:
;  1524          2099  2 !
;  1525          2100  2 !    This routine searches the device tables to verify the device
;  1526          2101  2 !    class and device name.
;  1527          2102  2 !
;  1528          2103  2 ! Calling Sequence:
;  1529          2104  2 !
;  1530          2105  2 !    TRANSLATE_CLASS (search_name,dev_class)
;  1531          2106  2 !
;  1532          2107  2 ! Input Parameters:
;  1533          2108  2 !
;  1534          2109  2 !    Search name = First two characters of device name
;  1535          2110  2 !
;  1536          2111  2 !    Dev_class = Device class to search for.
;  1537          2112  2 !
;  1538          2113  2 !
;  1539          2114  2 !    If the device class is found, then the specified device name
;  1540          2115  2 !    is compared against the device names in the device specific table.
;  1541          2116  2 !    Returns true if both match.
;  1542          2117  2 !
;  1543          2118  2 !    Returns false if device class and/or device name doesn't match.
;  1544          2119  2 !    (This should eventually be caught and handled by the parse_devname
;  1545          2120  2 !    routine.)
;  1546          2121  2 !
;  1547          2122  2 !--
;  1548          2123  2
;  1549          2124  2 EXTERNAL
;  1550          2125  2     Dev_addrs_ptr:      REF VECTOR [,long],
;  1551          2126  2     Dev_class_ptr:      REF VECTOR [,word],
;  1552          2127  2     Max_classes:        REF VECTOR [,byte];
;  1553          2128  2
;  1554          2129  2 OWN
;  1555          2130  2     I:          BYTE Initial (1),       ! Device address pointer index
;  1556          2131  2     Max_classes_value:  BYTE ;
;  1557          2132  2
;  1558          2133  2 LOCAL
;  1559          2134  2     Dev_specific_tbl:   REF VECTOR [,word], ! Device specific table address
;  1560          2135  2     K:          Initial (0) ;           ! Device specific table index
;  1561          2136  2
;  1562          2137  2 BIND
;  1563          2138  2     Cs_name = CH$PTR (uplit('CS')) ;
;  1564          2139  2
;  1565          2140  2 !
;  1566          2141  2 ! Device class ptr is the address of a table that contains supported device
;  1567          2142  2 ! classes and pointers to the device class specific information tables.
;  1568          2143  2 !
;  1569          2144  2 ! The device class specific table contains the supported device names,
;  1570          2145  2 ! image name pointers (image that needs to get activated), and transfer
;  1571          2146  2 ! address pointers.
;  1572          2147  2 !
;  1573          2148  2 ! This routine locates the matching device class retrieves the device
;  1574          2149  2 ! specific pointer and matches the specified device name against those
```

```
;  1575          2150  2 ! in the device specific table.
;  1576          2151  2 !
;  1577          2152  2 ! Loop through all of the device class entries.
;  1578          2153  2 !
;  1579          2154  2 Max_classes_value = max_classes[0] ;
;  1580          2155  2
;  1581          2156  2 Incr I from 1 to .max_classes_value do
;  1582          2157  3     Begin
;  1583          2158  3     If .dev_class_ptr[.I] EQL .dev_class
;  1584          2159  3     Then
;  1585          2160  4         Begin
;  1586          2161  4         !
;  1587          2162  4         ! Get the address of a device class specific table.
;  1588          2163  4         !
;  1589          2164  4         Dev_specific_tbl = .dev_addrs_ptr[.I] ;
;  1590          2165  4
;  1591          2166  4         !
;  1592          2167  4         ! Initialize another index for the device class specific table so don't
;  1593          2168  4         ! lose the current position. Determine if the contents of the device
;  1594          2169  4         ! name field is valid OR whether the end of the device name entries
;  1595          2170  4         ! in the table has been reached.
;  1596          2171  4         !
;  1597          2172  4         K = 1 ;
;  1598          2173  4         Until (.K EQL .dev_specific_tbl[0]) do
;  1599          2174  5             Begin
;  1600          2175  5             !
;  1601          2176  5             ! Determine if the selected device name matches any of the
;  1602          2177  5             ! device names recorded in this table.
;  1603          2178  5             !
;  1604          2179  5             If CH$EQL (2, CH$PTR(.search_name), 2, CH$PTR(dev_specific_tbl[.K]))
;  1605          2180  5             Then
;  1606          2181  5                 !
;  1607          2182  5                 ! The device names match. Using the class dir table index,
;  1608          2183  5                 ! get the corresponding device class.
;  1609          2184  5                 !
;  1610          2185  5                 Return true ;
;  1611          2186  5
;  1612          2187  5             !
;  1613          2188  5             ! Update the device name pointer indices.
;  1614          2189  5             !
;  1615          2190  5             K = .K + 1 ;
;  1616          2191  4             End ;
;  1617          2192  3         End ;
;  1618          2193  2     End ;
;  1619          2194  2
;  1620          2195  2
;  1621          2196  2 !
;  1622          2197  2 ! The name for the console device 'CSA' is not included in the device name
;  1623          2198  2 ! tables contained in ERFLIB.TLB. It really is a second device name for
;  1624          2199  2 ! the RX device which is included in the device tables. There should be
;  1625          2200  2 ! a table that includes devices like these, however because there is only
;  1626          2201  2 ! one at this time, it is checked for explicitly.
;  1627          2202  2 !
;  1628          2203  2 If CH$EQL (2, CH$PTR(.search_name), 2, cs_name)
;  1629          2204  2 Then
;  1630          2205  2     !
;  1631          2206  2     ! This is a 'CS' entry, determine whether the 'CS' device class
```

```
;  1632          2207  2      ! matches the device class being searched for.
;  1633          2208  2      !
;  1634          2209  2      Begin
;  1635          2210  2      If .dev_class EQL DC$_DISK
;  1636          2211  2      Then
;  1637          2212  2          ! Indicate that the device class matches by returning with
;  1638          2213  2          ! a true value.
;  1639          2214  2          !
;  1640          2215  2          Return true ;
;  1641          2216  2      End ;
;  1642          2217  2
;  1643          2218  2
;  1644          2219  2  !
;  1645          2220  2  ! Could not locate a class for this device name.
;  1646          2221  2  !
;  1647          2222  2  Return false ;
;  1648          2223  2
;  1649          2224  1  End ;           ! Routine


                                                        .PSECT  $PLIT,NOWRT,NOEXE,  PIC,2
                                  00  00  53  43  00008 P.AAC:  .ASCII  \CS\<0><0>                                  ;
                                                        .PSECT  $OWN$,NOEXE,  PIC,2
                                              01  00036 I:      .BYTE   1                                           ;
                                                  00037 MAX_CLASSES_VALUE.
                                                        .BLKB   1

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

                                                        .PSECT  $CODE,NOWRT,  PIC,2

                                                    003C 00000          .ENTRY  TRANSLATE_CLASS, Save R2,R3,R4,R5       ; 2093
                                      55 00000000'  00   9E 00002       MOVAB   MAX_CLASSES_VALUE, R5                   ;
                                                    52   D4 00009       CLRL    K                                       ; 2094
                                      65 00000000G  00   90 0000B       MOVB    MAX_CLASSES, MAX_CLASSES_VALUE          ; 2154
                                      54            65   9A 0001?       MOVZBL  MAX_CLASSES_VALUE, R4                   ; 2156
                                                    50   D4 00015       CLRL    I                                       ; 2158
                                                    32   11 00017       BRB     3$                                      ;
                                      51 00000000G  00   D0 00019 1$:    MOVL    DEV_CLASS_PTR, R1                       ;
                                                 6140   3F 00020       PUSHAW  (R1)[I]                                 ;
08  AC              9E                10            00   ED 00023       CMPZV   #0, #16, @(SP)+, DEV_CLASS              ;
                                                    20   12 00029       BNEQ    3$                                      ;
                                      51 00000000G  00   D0 0002B       MOVL    DEV_ADDRS_PTR, R1                       ; 2164
                                      53            6140 D0 00032       MOVL    (R1)[I], DEV_SPECIFIC_TBL               ;
                                      52                 01 D0 00036    MOVL    #1, K                                   ; 2172
    52            63                10                   00 ED 00039 2$:  CMPZV   #0, #16, (DEV_SPECIFIC_TBL), K          ; 2173
                                                         0B 13 0003E    BEQL    3$                                      ;
                                    6342        04       BC B1 00040    CMPW    @SEARCH_NAME, (DEV_SPECIFIC_TBL)[K]     ; 2179
                                                         18 13 00045    BEQL    4$                                      ;
                                                         52 D6 00047    INCL    K                                       ; 2190
                                                         EE 11 00049    BRB     2$                                      ; 2173
```

```
                    CA              50          54  F3 0004B 3$:     AOBLEQ  R4, 1, 1$                        ; 2156
                       00000000'    00      04  BC  B1 0004F         CMPW    @SEARCH_NAME, CS_NAME             ; 2203
                                                0A  12 00057         BNEQ    5$                                ;
                                    01      08  AC  D1 00059         CMPL    DEV_CLASS, #1                     ; 2210
                                                04  12 0005D         BNEQ    5$                                ;
                                    50          01  D0 0005F 4$:     MOVL    #1, R0                            ; 2215
                                                04 00062             RET                                       ;
                                                50  D4 00063 5$:     CLRL    R0                                ; 2222
                                                04 00065             RET                                       ; 2224

; Routine Size:  102 bytes,    Routine Base:  $CODE + 06A4

; 1650          2225  1
; 1651          2226  1
; 1652          2227  1 End
; 1653          2228  0 ELUDOM



                                                                     .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  $OWN$                                  56  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  $CODE                                 180  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  $PLIT                                  12  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)



;                                Library Statistics
;
;                                                 -------- Symbols --------      Pages       Processing
;       File                                      Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA18:[SYSLIB]LIB.L32;1                  18619        72         0        1000        00:01.4



;                                       COMMAND QUALIFIERS

.       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RECSELECT/OBJ=OBJ$:RECSELECT MSRC$:RECSELECT/UPDATE=(BUG$:RECSELECT)

; Size:         1802 code + 68 data bytes
; Run Time:       00:29.4
; Elapsed Time:   01:07.2
; Lines/CPU Min:   4551
; Lexemes/CPU-Min: 27826
; Memory Used:  352 pages
```

```
; Compilation Complete
```

0441 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

VAXARITH
LIS

EMULAT

ERFPROC1
MAP

VAXEMUL
MAP

ERF

ERF
MAP

REGSELECT
LIS

0442 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

F11BXQP
MAP

ACCESS
LIS

TA78DVDEP
LIS

CLENUP
LIS

CHARGEQ
LIS

F11X