K 6

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

```
_$

Va
--
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
```

```
**FILE**ID**RECSELECT                                   G  7
```

```
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
```

REC
V04

```
0 MODULE RECSELECT
0 (%TITLE 'Entry Validation'
0 IDENT = 'V04-000') =
0
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 ! FACILITY:  ERF, Error Log Report Generator
1 !
1 ! ABSTRACT:
1 !
1 !       This routine will determine if the previously read entry
1 !       meets user specified selection criteria.
1 !
1 ! ENVIRONMENT:
1 !
1 !       VAX/VMS operating system, user mode.
1 !
1 ! AUTHOR:  Sharon Reynolds,     CREATION DATE:  January 1983
1 !
1 ! Modified by:
1 !
1 !       V03-022 EAD0179         Elliott A. Drayton              6-Jul-1984
1 !               Obtain LSTLUN value from SYECOM.
1 !
1 !       V03-023 SAR0274         Sharon A. Reynolds      19-Jun-1984
1 !               - Added another check for device selection and entry
1 !               selection combinations to fix a bug with
1 !               /INC=(MF,VOLUME) and /INC=(TAPE,VOLUME).
1 !
1 !       V03-022 EAD0179         Elliott A. Drayton      23-May-1984
1 !               Correct the passing of the address of device name
1 !               in VERIFY_DEVICE.
```

```
1 !
1 !         V03-021 SAR0267          Sharon A. Reynolds      15-May-1984
1 !                 - Updated VERIFY_DEVICE to support longer device names.
1 !                 - Added check for unknown entry output to replace code
1 !                 that was previously removed.
1 !
1 !         V03-020 SAR0254          Sharon A. Reynolds      23-Apr-1984
1 !                 Added flag to /before check to stop execution when
1 !                 last entry found.
1 !
1 !         V03-019 EAD0151          Elliott A. Drayton      14-Apr-1984
1 !                 Fixed structure names in VERIFY_DEVICE.
1 !
1 !         V03-018 EAD0141          Elliott A. Drayton      12-Apr-1984
1 !                 Removed reference to EMBETDEF.
1 !
1 !         V03-017 SAR0248          Sharon A. Reynolds      10-Apr-1984
1 !                 Moved the unknown keyword tests to the verify entry
1 !                 routine so it would go through same tests as any
1 !                 other /include or /exclude entry selection.
1 !
1 !         V03-016 SAR0245          Sharon A. Reynolds      4-Apr-1984
1 !                 Added EMB$LOGMSP to device type entry table.
1 !
1 !         V03-015 EAD0119          Elliott A. Drayton      23-Mar-1984
1 !                 Remove support for /UNKNOWN qualifier and added support
1 !                 for the UNKNOWN keyword.
1 !
1 !         V03-014 EAD0115          Elliott A. Drayton       9-Mar-1984
1 !                 Removed emb_buf and syecom_buf.
1 !
1 !         V03-013 SAR0189          Sharon A. Reynolds,     13-Feb-1984
1 !                 - Added 'CS' device name support to device table search
1 !                 routine.
1 !                 - Added additional test for entry summary update.
1 !
1 !         V03-012 SAR0184          Sharon A. Reynolds,     17-Jan-1984
1 !                 - Fixed a bug in the output of the erf_unkentry message.
1 !                 - Added code to set the end value indicator when
1 !                   the last selected entry (/entry) is found.
1 !
1 !         V03-011 SAR0181          Sharon A. Reynolds,     13-Dec-1983
1 !                 - Remove descriptor references.
1 !                 - Add device attention keyword support.
1 !                 - Add lm/sp entries to device errors entry list.
1 !                 - Add lm/sp entry check for bus class selections.
1 !                 - Removed logmessage keyword.
1 !                 - Add unsolicited_mscp keyword support.
1 !                 - Added incomplete entry message.
1 !
1 !         V03-010 SAR0176          Sharon A. Reynolds,     21-Nov-1983
1 !                 - Removed un-necessary check for outputting all
1 !                   entries.
1 !                 - Changed reference to report type.
1 !
1 !         V03-009 SAR0152          Sharon A. Reynolds,     7-Oct-1983
1 !                 - Added code to output informational messages when
```

```
1 !                     and unknown entry is encountered.
1 !                   - Added the code that counts intervening logmessage
1 !                     logstatus entries.
1 !                   - Re-structured the /include and /exclude entry
1 !                     checks to fix a bug.
1 !                   - Made /includ=disks/exclude=db1 a valid command.
1 !
1 !           V03-008 SAR0139         Sharon A. Reynolds,     20-Sep-1983
1 !                   Fixed a bug in mount/dismount output. Fixed an out
1 !                   of range loop.
1 !
1 !           V03-007 SAR0122         Sharon A. Reynolds,     23-Aug-1983
1 !                   Re-wrote translate_class routine for use with the
1 !                   permanent device tables.
1 !
1 !           V03-006 SAR0032         Sharon A. Reynolds,     2-Jun-1983
1 !                   Replaced emb_stuf with emb_buf definitions. Fixed bug
1 !                   in dc$_bus selection.
1 !
1 !           V03-005 SAR0029         Sharon A. Reynolds,     11-May-1983
1 !                   Removed support for logstatus keyword.
1 !
1 !           V03-004 SAR0013         Sharon A. Reynolds,     18-Apr-1983
1 !                   Deleted the log message and status message entries
1 !                   from the 'control' table. Added call to update
1 !                   entry summaries.
1 !
1 !           V03-003 SAR0003         Sharon A. Reynolds,     5-Apr-1983
1 !                   Removed the volume_output flag definition. Changed
1 !                   any references to volume_output flag so they refer
1 !                   to it from SYECOM.
1 !
1 !           V03-002 SAR0002         Sharon A. Reynolds,     5-Apr-1983
1 !                   Fixed /exclude selection bug.
1 !
1 !           V03-001 SAR0001         Sharon A. Reynolds,     29-Mar-1983
1 !                   Fixed /include='device name', volume mount/dismount
1 !                   selection problem.
1 !
1 !--
1
1
1 !
1 ! Required files
1 !
1 REQUIRE 'SRC$:ERFDEF.REQ' ;             ! ERF defintions
1 REQUIRE 'LIB$:PARSERDAT.R32' ;          ! ERF parser data definitions
1 REQUIRE 'SRC$:RECSELDEF.REQ' ;          ! EMB, SYECOM, LOGMSG, LOGSTS, and
1                                         !  VOLMOUNT field defintions
1
1 !
1 ! Table of contents
1 !
1 FORWARD ROUTINE
1     Record_selected,                    ! Verify entry against selections
1     Verify_entry,                       ! Verify the entry type
```

```
1	    Device_type_entry,                          ! Determine if it's a device type entry
1	    Verify_device_class,                        ! Verify the device class
1	    Verify_device,                              ! Verify the device name
1	    Translate_class ;                           ! Translate device class to a name
1
1 !
1 ! Declare external routines
1 !
1 EXTERNAL ROUTINE
1	    Exec_image,                                 ! Execute an image
1	    Intervene_increment,
1	    Intervene_output,
1	    Search_queue: addressing_mode (general) , ! Search queue of devices selected
1	    Validate_packet;                            ! Is the packet validate for the cpu it was logged on.
1
1 !
1 ! Declare external literals
1 !
1 EXTERNAL LITERAL
1	    Erf_incentry,
1	    Erf_unkclass,
1	    Erf_unkcpu,
1	    Erf_unkentry,
1	    Erf_unktype ;
1
1 !
1 ! Declare external data.
1 !
1 EXTERNAL
1	    Class_dir:          REF $BBLOCK,
1	    Device_class,
1	    Device_type,
1	    Emb:                $BBLOCK PSECT (EMB),
1	    Exclude_flag,
1	    Exclude_mask:       REF $BBLOCK,
1	    Include_mask:       REF $BBLOCK,
1	    Option_flag:        REF $BBLOCK,
1	    Parser_data:        REF $BBLOCK,
1	    Processor_type,
1	    Summary_dispatcher_addr,
1	    Summary_flag:       REF $BBLOCK,
1	    Syecom:             $BBLOCK PSECT (SYECOM),
1	    Unknown_entry ;
1
1 !
1 ! Declare literal definitions
1 !
1 LITERAL
1	    Incomplete_entry = 128 ;
1
1 !
1 ! Own storage definitions
1 !
1 OWN
1	    Lstlun.             Long,
1	    Dev_selection_required: BYTE,
1	    Device_status:      BYTE,
;  229      0797  1      Dev_cls_status:     BYTE,
;  230      0798  1      Dev_type_entry_sts: BYTE,
;  231      0799  1      Entry_status:       BYTE,
;  232      0800  1      Validate_pkt_sts:   Initial (false),
;  233      0801  1      Bugchks:            VECTOR [3,byte,unsigned]  ! Bugcheck type entries
;  234      0802  1                          Initial (byte
;  235      0803  1                                  (EMB$K_CR,      ! Crash
;  236      0804  1                                  EMB$K_SBC,      ! System bugchecks
;  237      0805  1                                  EMB$K_UBC)),    ! User bugchecks
;  238      0806  1
;  239      0807  1      Control:            VECTOR [7,byte,unsigned] ! Control type entries
;  240      0808  1                          Initial (byte
;  241      0809  1                                  (EMB$K_CS,      ! Cold re-start
;  242      0810  1                                  EMB$K_NF,       ! New file created
;  243      0811  1                                  EMB$K_WS,       ! Warm re-start
;  244      0812  1                                  EMB$K_TS,       ! Time stamp
;  245      0813  1                                  EMB$K_SS,       ! System service message
;  246      0814  1                                  EMB$K_OM,       ! Operator message
;  247      0815  1                                  EMB$K_NM)),     ! Network message
;  248      0816  1
;  249      0817  1      Cpu:                VECTOR [8,byte,unsigned] ! Cpu type entries
;  250      0818  1                          Initial (byte
;  251      0819  1                                  (EMB$K_AW,      ! Asynchronouw write error
;  252      0820  1                                  EMB$K_UBA,      ! Unibus adapter error
;  253      0821  1                                  EMB$K_MBA,      ! Massbus adapter error
;  254      0822  1                                  EMB$K_UI,       ! Undefined interrupt
;  255      0823  1                                  EMB$K_BE,       ! Bus error
;  256      0824  1                                  EMB$K_SA,       ! SBI alert
;  257      0825  1                                  EMB$K_SI,       ! 11/750 fault thru SBI vector
;  258      0826  1                                  EMB$K_UE)),     ! 11/730 unibus error
;  259      0827  1
;  260      0828  1      Dev_errors:         VECTOR [3,byte,unsigned] ! Device error entries
;  261      0829  1                          Initial (byte
;  262      0830  1                                  (EMB$K_DE,      ! Device Errors
;  263      0831  1                                  EMB$K_SP,       ! Logstatus entries (mscp)
;  264      0832  1                                  EMB$K_LM)),     ! Logmessage entries (mscp)
;  265      0833  1
;  266      0834  1      Memorys:            VECTOR [2,byte,unsigned] ! Memory entries
;  267      0835  1                          Initial (byte
;  268      0836  1                                  (EMB$K_SE,      ! Soft ECC error
;  269      0837  1                                  EMB$K_HE)),     ! Hard ECC error
;  270      0838  1
;  271      0839  1      Volume:             VECTOR [2,byte,unsigned]        ! Volume change entries
;  272      0840  1                          Initial (byte
;  273      0841  1                                  (EMB$K_VM,      ! Volume mounts
;  274      0842  1                                  EMB$K_VD)) ;    ! Volume dismounts
;  275      0843  1
```

```
1 GLOBAL ROUTINE RECORD_SELECTED =
2 Begin
2
2 !++
2 !
2 ! Functional Description:
2 !
2 !       This routine will determine what selection qualifiers are
2 !       specified and match the appropriate fields in the current
2 !       entry against the selections. It return TRUE if the
2 !       current entry matches or return FALSE if the current entry
2 !       does NOT match.
2 !
2 ! Calling sequence:
2 !
2 !       RECORD_SELECTED ()
2 !
2 ! Input parameters:
2 !
2 !       None
2 !
2 ! Output parameters:
2 !
2 !       None
2 !
2 !--
2
2 LOCAL
2     Include_status:     BYTE
2                         Initial (true),
2     Exclude_status:     BYTE
2                         Initial (true) ;
2
2 lstlun = .syecom [sye$l_lstlun];
2
2 !
2 ! Validate the packet for entry/cpu type and device class/type.
2 !
3 If NOT (VALIDATE_PACKET ())
2 Then
2     Unknown_entry = true
2 Else
2     Unknown_entry = false ;
2
2 !
2 ! Determine if /summary selected and update that entry summary
2 ! information.
2 !
3 If (.option_flag[opt$v_summary_qual] AND
4     (.summary_flag[sum$v_entry] OR
4      .summary_flag[sum$v_all_summ] OR
3      .summary_flag[sum$v_histogram]))
2 Then
2     Exec_image (summary_dispatcher_addr,lstlun,%REF(entry_summ_upd)) ;
2
2 !
2 ! If incomplete entry report the error.
```

```
2 !
3 If ((NOT .syecom[sye$b_valid_entry]) AND
3    (.emb[emb$w_hd_entry] GEQ incomplete_entry))
2 Then
3    Begin
3    Signal (erf_incentry, 1, .emb[emb$w_hd_entry]);
3    Return false;
2    End;
2
2 !
2 ! Determine whether the volume mounts/dismounts should be output or just
2 ! label information saved from the entry.
2 !
3 If (.exclude_mask[exc$v_volume] AND
4     (.include_mask[inc$v_device_select] OR
4     .include_mask[inc$v_dev_class_select] OR
4     .include_mask[inc$v_dev_attentions] OR
4     .include_mask[inc$v_dev_errors] OR
2     .include_mask[inc$v_dev_timeouts])) AND
3     (NOT .include_mask[inc$v_volume] OR
3     NOT .option_flag[opt$v_output_all])
2 Then
2     !
2     ! Indicate that volume mount/dismount entries
2     ! should not be output.
2     !
2     Syecom[sye$b_volume_output] = false
2 Else
2     Syecom[sye$b_volume_output] = true ;
2
2 !
2 ! Determine if the /ENTRY qualifier was specified.
2 !
2 If .option_flag[opt$v_entry_qual]
2 Then
2     !
2     ! /Entry specified, get the address of the entry selection
2     ! data and determine if the number of this entry
2     ! is within the selected range.
2     !
3     Begin
3     If .syecom[sye$l_reccnt] LSSU .parser_data[erl$l_end_entry]
3     Then
3         !
3         ! This entry should be within the selected range, ensure
3         ! the entry number is greater than the starting entry selection.
3         !
4         Begin
5         If NOT (.syecom[sye$l_reccnt] GEQU .parser_data[erl$l_start_entry])
4         Then
4             !
4             ! Entry is NOT within the selected range, return to calling
4             ! routine.
4             !
4             Return false ;
4         End
3     Else
```

```
;  391        0958  3            !
;  392        0959  3            ! Entry is NOT within the selected range, return to calling
;  393        0960  3            ! routine.
;  394        0961  3            !
;  395        0962  4            Begin
;  396        0963  4            If .syecom[sye$l_reccnt] GEQU .parser_data[erl$l_end_entry]
;  397        0964  4            Then
;  398        0965  4                !
;  399        0966  4                ! Indicate that last selected entry was found.
;  400        0967  4                !
;  401        0968  4                Syecom[sye$b_end_value] = true ;
;  402        0969  4
;  403        0970  4            Return true ;
;  404        0971  3            End ;
;  405        0972  2        End ;
;  406        0973  2
;  407        0974  2  !
;  408        0975  2  ! Determine if the /BEFORE qualifier was specified.
;  409        0976  2  !
;  410        0977  2  If .option_flag[opt$v_before_qual]
;  411        0978  2  Then
;  412        0979  2        !
;  413        0980  2        ! Determine if the date/time that this entry was recorded falls
;  414        0981  2        ! within the range of selected date/times.
;  415        0982  2        !
;  416        0983  3        Begin
;  417      P 0984  3        If COMPARE_QUAD(emb[emb$q_hd_time],GEQU,
;  418        0985  4                        parser_data[erl$q_end_date])
;  419        0986  3        Then
;  420        0987  3            !
;  421        0988  3            ! This entry is NOT within the selected date/time range,
;  422        0989  3            ! return to the calling routine.
;  423        0990  3            !
;  424        0991  4            Begin
;  425        0992  4            Syecom[sye$b_end_value] = true ;
;  426        0993  4            Return true ;
;  427        0994  3            End ;
;  428        0995  2        End ;
;  429        0996  2
;  430        0997  2  !
;  431        0998  2  ! Determine if the /SINCE qualifier was specified.
;  432        0999  2  !
;  433        1000  2  If .option_flag[opt$v_since_qual]
;  434        1001  2  Then
;  435        1002  2        !
;  436        1003  2        ! Ensure that the entry date/time is greater than the starting
;  437        1004  2        ! time/date selection.
;  438        1005  2        !
;  439        1006  3        Begin
;  440      P 1007  3        If NOT COMPARE_QUAD(emb[emb$q_hd_time],GEQU,
;  441        1008  4                            parser_data[erl$q_start_date])
;  442        1009  3        Then
;  443        1010  3            !
;  444        1011  3            ! The entry does NOT meet that selection criteria for date/time,
;  445        1012  3            ! return to the calling routine.
;  446        1013  3            !
;  447        1014  3            Return false ;
```

```
2       End ;
2
2 !
2 ! Determine if the /SID_REGISTER qualifier was specified.
2 !
2 If .option_flag[opt$v_sid_reg_qual]
2 Then
2         !
2         ! Determine if the entry sid matches the selected sid.
2         !
3       Begin
3       If NOT .parser_data[erl$l_sid_selection] EQLU .emb[emb$l_hd_sid]
3       Then
3           !
3           ! Entry sid does NOT match selected sid, return to calling
3           ! routine.
3           !
3           Return false ;
2       End ;
2
2 Device_status = false ;
2 Dev_cls_status = false ;
2 Entry_status = false ;
2
2 Dev_type_entry_sts = DEVICE_TYPE_ENTRY () ;
2
2 If .option_flag[opt$v_include_qual]
2 Then
3       Begin
3       Exclude_flag = false ;
3
3       If .dev_type_entry_sts OR
3           (.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
4           (.emb[emb$w_hd_entry] EQLU EMB$K_VD)
3       Then
4           Begin
4           If .include_mask[inc$v_device_select]
4           Then
5               Begin
5               If VERIFY_DEVICE ()
5               Then
5                   Device_status = true
5               Else
5                   Device_status = false ;
4               End ;
4
4           If .include_mask[inc$v_dev_class_select]
4           Then
5               Begin
5               If VERIFY_DEVICE_CLASS ()
5               Then
5                   Dev_cls_status = true
5               Else
5                   Dev_cls_status = false ;
4               End ;
3           End ;
3
```

```
3       If .include_mask[inc$v_entry_select]
3       Then
4           Begin
4           If VERIFY_ENTRY ()
4           Then
4               Entry_status = true
4           Else
4               Entry_status = false ;
3           End ;
3
3
4       If (.include_mask[inc$v_device_select] AND
3           .dev_type_entry_sts AND .device_status) OR
3
4           (.include_mask[inc$v_dev_class_select] AND
3           .dev_type_entry_sts AND .dev_cls_status) OR
3
4           (.include_mask[inc$v_entry_select] AND .entry_status)
3       Then
3           Include_status = true
3       Else
3           Include_status = false ;
3
3
3       If .include_mask[inc$v_device_select] AND
3           .include_mask[inc$v_entry_select]
3       Then
4           Begin
4           Include_status = false ;
4
4           If .dev_selection_required
4           Then
5               Begin
5               If (.entry_status AND .device_status) OR
6                   (.dev_type_entry_sts AND .device_status)
5               Then
5                   Include_status = true ;
5               End
4           Else
5               Begin
5               If .dev_type_entry_sts AND .device_status
5               Then
6                   Begin
6                   Include_status = true ;
6                   End
5               Else
6                   Begin
7                   If (NOT .dev_type_entry_sts AND .entry_status)
6                   Then
6                       Include_status = true ;
5                   End ;
4               End ;
3           End ;
3
3       If .include_mask[inc$v_dev_class_select] AND
3           .include_mask[inc$v_entry_select]
3       Then
```

```
;  4          Begin
;  4          Include_status = false ;
;  4
;  4          If .dev_selection_required
;  4          Then
;  5              Begin
;  5              If (.entry_status AND .dev_cls_status) OR
;  6                  (.dev_type_entry_sts AND .dev_cls_status)
;  5              Then
;  5                  Include_status = true ;
;  5              End
;  4          Else
;  5              Begin
;  5              If .dev_type_entry_sts AND .dev_cls_status
;  5              Then
;  6                  Begin
;  6                  Include_status = true ;
;  6                  End
;  5              Else
;  6                  Begin
;  7                  If (NOT .dev_type_entry_sts AND .entry_status)
;  6                  Then
;  6                      Include_status = true ;
;  5                  End ;
;  4              End ;
;  3          End ;
;  3
;  2      End ;
;  2
;  2  !
;  2  !If not /include option then include_status = false
;  2  !
;  2
;  2  If .option_flag[opt$v_exclude_qual]
;  2  Then
;  3      Begin
;  3      Exclude_flag = true ;
;  3
;  3      If .dev_type_entry_sts OR
;  3          (.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
;  4          (.emb[emb$w_hd_entry] EQLU EMB$K_VD)
;  3      Then
;  4          Begin
;  4          If .exclude_mask[exc$v_device_select]
;  4          Then
;  5              Begin
;  5              If VERIFY_DEVICE ()
;  5              Then
;  5                  Device_status = true
;  5              Else
;  5                  Device_status = false ;
;  4              End ;
;  4
;  4          If .exclude_mask[exc$v_dev_class_select]
;  4          Then
;  5              Begin
;  5              If VERIFY_DEVICE_CLASS ()
```

```
5                   Then
5                       Dev_cls_status = true
5                   Else
5                       Dev_cls_status = false ;
4                   End ;
3               End ;
3
3           If .exclude_mask[exc$v_entry_select]
3           Then
4               Begin
4               If VERIFY_ENTRY ()
4               Then
4                   Entry_status = true
4               Else
4                   Entry_status = false ;
3               End ;
3
4           If (.exclude_mask[exc$v_device_select] AND
3               .dev_type_entry_sts AND .device_status) OR
3
4               (.exclude_mask[exc$v_dev_class_select] AND
3               .dev_type_entry_sts AND .dev_cls_status) OR
3
4               (.exclude_mask[exc$v_entry_select] AND .entry_status)
3           Then
3               Exclude_status = false
3           Else
3               Exclude_status = true ;
3
3
3           If .exclude_mask[exc$v_device_select] AND
3               .exclude_mask[exc$v_entry_select]
3           Then
4               Begin
4               Exclude_status = true ;
4
4               If .dev_selection_required
4               Then
5                   Begin
5                   If (.entry_status AND .device_status) OR
6                       (.dev_type_entry_sts AND .device_status)
5                   Then
5                       Exclude_status = false ;
5                   End
4               Else
5                   Begin
5                   If .dev_type_entry_sts AND .device_status
5                   Then
6                       Begin
6                       Exclude_status = false ;
6                       End
5                   Else
6                       Begin
7                       If (NOT .dev_type_entry_sts AND .entry_status)
6                       Then
6                           Exclude_status = false ;
5                       End ;
```

```
;   4                  End ;
;   3              End ;
;   3
;   3          If .exclude_mask[exc$v_dev_class_select] AND
;   3              .exclude_mask[exc$v_entry_select]
;   3          Then
;   4              Begin
;   4              Exclude_status = true ;
;   4
;   4              If .dev_selection_required
;   4              Then
;   5                  Begin
;   5                  If (.entry_status AND .dev_cls_status) OR
;   6                      (.dev_type_entry_sts AND .dev_cls_status)
;   5                  Then
;   5                      Exclude_status = false ;
;   5                  End
;   4              Else
;   5                  Begin
;   5                  If .dev_type_entry_sts AND .dev_cls_status
;   5                  Then
;   6                      Begin
;   6                      Exclude_status = false ;
;   6                      End
;   5                  Else
;   6                      Begin
;   7                      If (NOT .dev_type_entry_sts AND .entry_status)
;   6                      Then
;   6                          Exclude_status = false ;
;   5                      End ;
;   4                  End ;
;   3              End ;
;   3
;   2          End ;       ! of /exclude processing
;   2      !
;   2      ! If /exclude option match, exclude_status = false.
;   2      !
;   2
;   2
;   2      !
;   2      ! Determine whether to count logmessage/logstatus entries.
;   2      !
;   3      If ( (.include_status) AND (.exclude_status) AND
;   3          (.parser_data[erl$b_rpt_type] EQL full_rep) )
;   2      Then
;   2          !
;   2          ! Determine if it was a logmessage/logstatus entry.
;   2          !
;   3          Begin
;   3          If (.emb[emb$w_hd_entry] EQLU EMB$C_SP) OR
;   4              (.emb[emb$w_hd_entry] EQLU EMB$C_LM)
;   3          Then
;   3              !
;   3              ! Count the number of logmessage/logstatus entries
;   3              ! that might be skipped.
;   3              !
;   3              INTERVENE_INCREMENT (lstlun)
```

```
3      Else
3          !
3          ! Determine whether to output the logstatus/logmessage
3          ! intervening message and if necessary output it.
3          !
3          INTERVENE_OUTPUT (lstlun) ;
2      End ;
2
2 !
2 ! Determine if the entry met the selection criteria.
2 !
2 ! Determine if this is an unknown entry.
2 !
2 If .unknown_entry
2 Then
2     ! Indicate that this is an unknown entry and return with a
2     ! true value so that it will be output.
2     !
2     Return true ;
2
2 If (NOT .include_status) OR
3     (NOT .exclude_status)
2 Then
2     !
2     ! Indicate that the entry should not be output by
2     ! returning to the calling routine with a false value.
2     !
2     Return false ;
2
2 !
2 ! Indicate that the entry should be output by
2 ! returning to the calling routine with a true value.
2 !
2 Return true ;
2
1 End ;   ! Routine


                                     .TITLE  RECSELECT Entry Validation
                                     .IDENT  \V04-000\

                                     .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                              ;
                                             00013          .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_UNKENTRY
                                                            .EXTRN  ERF_UNKTYPE, CLASS_DIR
                                                            .EXTRN  DEVICE_CLASS, DEVICE_TYPE
                                                            .EXTRN  EMB, EXCLUDE_FLAG
                                                            .EXTRN  EXCLUDE_MASK, INCLUDE_MASK
                                                            .EXTRN  OPTION_FLAG, PARSER_DATA
                                                            .EXTRN  PROCESSOR_TYPE, SUMMARY_DISPATCHER_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,-; 0844
                                                                    R9,R10,R11
                    5B 00000000G  00  9E  00002          MOVAB   PARSER_DATA, R11                         ;
                    5A 00000000G  00  9E  00009          MOVAB   EXCLUDE_MASK, R10                        ;
                    59 00000000G  00  9E  00010          MOVAB   OPTION_FLAG, R9                          ;
                    58 00000000G  00  9E  00017          MOVAB   SYECOM+24, R8                            ;
                    57 00000000G  00  9E  0001E          MOVAB   INCLUDE_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                       ; 0845
                    53            01  90  00039          MOVB    #1, EXCLUDE_STATUS                       ;
               F8   A5       0F   A8  D0  0003C          MOVL    SYECOM+39, LSTLUN                        ; 0877
        00000000G   00            00  FB  00041          CALLS   #0, VALIDATE_PACKET                      ; 0882
                    09            50  E8  00048          BLBS    R0, 1$                                   ;
        00000000G   00            01  D0  0004B          MOVL    #1, UNKNOWN_ENTRY                        ; 0884
                                  06  11  00052          BRB     2$                                       ;
                       00000000G  00  D4  00054 1$:      CLRL    UNKNOWN_ENTRY                            ; 0886
                    50            69  D0  0005A 2$:      MOVL    OPTION_FLAG, R0                          ; 0892
      27            60            0E  E1  0005D          BBC     #14, (R0), 4$                            ;
                    50 00000000G  00  D0  00061          MOVL    SUMMARY_FLAG, R0                         ; 0893
      07            60            02  E0  00068          BBS     #2, (R0), 3$                             ;
                    04            60  E8  0006C          BLBS    (R0), 3$                                 ; 0894
      15            60            05  E1  0006F          BBC     #5, (R0), 4$                             ; 0895
                    6E            05  D0  00073 3$:      MOVL    #5, (SP)                                 ; 0897
                                  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                           ;
```

```
                                                           J  8
RECSELECT          Entry Validation                   15-Sep-1984 23:52:05    VAX-11 Bliss-32 V4.0-742           Page  16
V04-000                                               14-Sep-1984 12:28:02    [ERF.SRC]RECSELECT.B32;1                 (2)

                            1C        03  A8  E8 00088 4$:    BLBS    SYECOM+27, 6$                     ; 0902
                  0080      8F            66  B1 0008C        CMPW    EMB+4, #128                       ; 0903
                                          15  1F 00091        BLSSU   6$                                ;
                            7E            66  3C 00093        MOVZWL  EMB+4, -(SP)                      ; 0906
                                          01  DD 00096        PUSHL   #1                                ;
                                00000000G 8F  DD 00098        PUSHL   #ERF_INCENTRY                     ;
             00000000G      00            03  FB 0009E        CALLS   #3, LIB$SIGNAL                    ;
                                        02E9  31 000A5 5$:    BRW     69$                               ; 0907
                            50            6A  D0 000A8 6$:    MOVL    EXCLUDE_MASK, R0                  ; 0914
          29                60            12  E1 000AB        BBC     #18, (R0), 9$                     ;
                            50            67  D0 000AF        MOVL    INCLUDE_MASK, R0                  ; 0915
          10                60            14  E0 000B2        BBS     #20, (R0), 7$                     ;
          0C                60            15  E0 000B6        BBS     #21, (R0), 7$                     ; 0916
          08                60            09  E0 000BA        BBS     #9, (R0), 7$                      ; 0917
          04                60            0D  E0 000BE        BBS     #13, (R0), 7$                     ; 0918
                            12        02  A0  E9 000C2        BLBC    2(R0), 9$                         ; 0919
                            50            67  D0 000C6 7$:    MOVL    INCLUDE_MASK, R0                  ; 0920
          07                60            12  E1 000C9        BBC     #18, (R0), 8$                     ;
                            50            69  D0 000CD        MOVL    OPTION_FLAG, R0                   ; 0921
                                          60  B5 000D0        TSTW    (R0)                              ;
                                          04  19 000D2        BLSS    9$                                ;
                                          68  94 000D4 8$:    CLRB    SYECOM+24                         ; 0927
                                          03  11 000D6        BRB     10$                               ;
                            68            01  90 000D8 9$:    MOVB    #1, SYECOM+24                     ; 0929
                            52            69  D0 000DB 10$:   MOVL    OPTION_FLAG, R2                   ; 0934
          13                62            03  E1 000DE        BBC     #3, (R2), 11$                     ;
                            51        E8  A8  D0 000E2        MOVL    SYECOM, R1                        ; 0942
                            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)                        ; 0949
                                          B0  1F 000F3        BLSSU   5$                                ;
                            1B            62  E9 000F5 11$:   BLBC    (R2), 14$                         ; 0977
          50                6B            05  C1 000F8        ADDL3   #5, PARSER_DATA, R0               ; 0985
                            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      A8            01  90 0010C 13$:   MOVB    #1, SYECOM+30                     ; 0992
                                        027A  31 00110        BRW     68$                               ; 0993
          18                62            0D  E1 00113 14$:   BBC     #13, (R2), 16$                    ; 1000
          50                6B            0D  C1 00117        ADDL3   #13, PARSER_DATA, R0              ; 1008
                            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$:   BBC     #12, (R2), 18$                    ; 1020
                            50            6B  D0 00133        MOVL    PARSER_DATA, R0                   ; 1026
                    FC      A6        01  A0  D1 00136        CMPL    1(R0), EMB                        ;
                                          03  13 0013B        BEQL    18$                               ;
                                        0251  31 0013D 17$:   BRW     69$                               ;
                                      FD  A5  B4 00140 18$:   CLRW    DEVICE_STATUS                     ; 1035
                                          65  94 00143        CLRB    ENTRY_STATUS                      ; 1037
```

```
   00000000V  00           00 FB 00145         CALLS   #0, DEVICE_TYPE_ENTRY                   ; 1039
          FF  A5           50 90 0014C         MOVB    R0, DEV_TYPE_ENTRY_STS                  ;
              50           69 D0 00150         MOVL    OPTION_FLAG, R0                         ; 1041
03            60           06 E0 00153         BBS     #6, (R0), 19$                           ;
                         00F3 31 00157         BRW     41$                                     ;
                00000000G  00 D4 0015A 19$:    CLRL    EXCLUDE_FLAG                            ; 1044
              0E       FF  A5 E8 00160         BLBS    DEV_TYPE_ENTRY_STS, 20$                 ; 1046
        0040  8F           66 B1 00164         CMPW    EMB+4, #64                              ; 1047
                           07 13 00169         BEQL    20$                                     ;
        0041  8F           66 B1 0016B         CMPW    EMB+4, #65                              ; 1048
                           34 12 00170         BNEQ    24$                                     ;
              50           67 D0 00172 20$:    MOVL    INCLUDE_MASK, R0                        ; 1051
13            60           14 E1 00175         BBC     #20, (R0), 22$                          ;
   00000000V  00           00 FB 00179         CALLS   #0, VERIFY_DEVICE                       ; 1054
              06           50 E9 00180         BLBC    R0, 21$                                 ;
          FD  A5           01 90 00183         MOVB    #1, DEVICE_STATUS                       ; 1056
                           03 11 00187         BRB     22$                                     ;
                       FD  A5 94 00189 21$:    CLRB    DEVICE_STATUS                           ; 1058
              50           67 D0 0018C 22$:    MOVL    INCLUDE_MASK, R0                        ; 1061
13            60           15 E1 0018F         BBC     #21, (R0), 24$                          ;
   00000000V  00           00 FB 00193         CALLS   #0, VERIFY_DEVICE_CLASS                 ; 1064
              06           50 E9 0019A         BLBC    R0, 23$                                 ;
          FE  A5           01 90 0019D         MOVB    #1, DEV_CLS_STATUS                      ; 1066
                           03 11 001A1         BRB     24$                                     ;
                       FE  A5 94 001A3 23$:    CLRB    DEV_CLS_STATUS                          ; 1068
              50           67 D0 001A6 24$:    MOVL    INCLUDE_MASK, R0                        ; 1072
11            60           16 E1 001A9         BBC     #22, (R0), 26$                          ;
   00000000V  00           00 FB 001AD         CALLS   #0, VERIFY_ENTRY                        ; 1075
              05           50 E9 001B4         BLBC    R0, 25$                                 ;
              65           01 90 001B7         MOVB    #1, ENTRY_STATUS                        ; 1077
                           02 11 001BA         BRB     26$                                     ;
                           65 94 001BC 25$:    CLRB    ENTRY_STATUS                            ; 1079
              50           67 D0 001BE 26$:    MOVL    INCLUDE_MASK, R0                          1083
08            60           14 E1 001C1         BBC     #20, (R0), 27$                          ;
              04       FF  A5 E9 001C5         BLBC    DEV_TYPE_ENTRY_STS, 27$                 ; 1084
              13       FD  A5 E8 001C9         BLBS    DEVICE_STATUS, 29$                      ;
08            60           15 E1 001CD 27$:    BBC     #21, (R0), 28$                          ; 1086
              04       FF  A5 E9 001D1         BLBC    DEV_TYPE_ENTRY_STS, 28$                 ; 1087
              07       FE  A5 E8 001D5         BLBS    DEV_CLS_STATUS, 29$                     ;
08            60           16 E1 001D9 28$:    BBC     #22, (R0), 30$                          ; 1089
              05           65 E9 001DD         BLBC    ENTRY_STATUS, 30$                       ;
              54           01 90 001E0 29$:    MOVB    #1, INCLUDE_STATUS                      ; 1091
                           02 11 001E3         BRB     31$                                     ;
                           54 94 001E5 30$:    CLRB    INCLUDE_STATUS                          ; 1093
2F            60           14 E1 001E7 31$:    BBC     #20, (R0), 36$                          ; 1096
2B            60           16 E1 001EB         BBC     #22, (R0), 36$                          ; 1097
                           54 94 001EF         CLRB    INCLUDE_STATUS                          ; 1100
              11       FC  A5 E9 001F1         BLBC    DEV_SELECTION_REQUIRED, 33$             ; 1102
              04           65 E9 001F5         BLBC    ENTRY_STATUS, 32$                       ; 1105
              1B       FD  A5 E8 001F8         BLBS    DEVICE_STATUS, 35$                      ;
              1A       FF  A5 E9 001FC 32$:    BLBC    DEV_TYPE_ENTRY_STS, 36$                 ; 1106
              16       FD  A5 E9 00200         BLBC    DEVICE_STATUS, 36$                      ;
                           11 11 00204         BRB     35$                                     ; 1108
              51       FF  A5 9A 00206 33$:    MOVZBL  DEV_TYPE_ENTRY_STS, R1                  ; 1112
              07           51 E9 0020A         BLBC    R1, 34$                                 ;
              06       FD  A5 E8 0020D         BLBS    DEVICE_STATUS, 35$                      ;
              06           51 E8 00211         BLBS    R1, 36$                                 ; 1119
```

```
                03        65 E9 00214 34$:   BLBC    ENTRY_STATUS, 36$                    ;
                54        01 90 00217 35$:   MOVB    #1, INCLUDE_STATUS                    ; 1121
     2F         60        15 E1 0021A 36$:   BBC     #21, (R0), 41$                        ; 1126
     2B         60        16 E1 0021E        BBC     #22, (R0), 41$                        ; 1127
                          54 94 00222        CLRB    INCLUDE_STATUS                        ; 1130
                11     FC A5 E9 00224        BLBC    DEV_SELECTION_REQUIRED, 38$           ; 1132
                04        65 E9 00228        BLBC    ENTRY_STATUS, 37$                     ; 1135
                1B     FE A5 E8 0022B        BLBS    DEV_CLS_STATUS, 40$                   ;
                1A     FF A5 E9 0022F 37$:   BLBC    DEV_TYPE_ENTRY_STS, 41$               ; 1136
                16     FE A5 E9 00233        BLBC    DEV_CLS_STATUS, 41$                   ;
                          11 11 00237        BRB     40$                                   ; 1138
                50     FF A5 9A 00239 38$:   MOVZBL  DEV_TYPE_ENTRY_STS, R0                ; 1142
                07        50 E9 0023D        BLBC    R0, 39$                               ;
                06     FE A5 E8 00240        BLBS    DEV_CLS_STATUS, 40$                   ;
                06        50 E8 00244        BLBS    R0, 41$                               ; 1149
                03        65 E9 00247 39$:   BLBC    ENTRY_STATUS, 41$                     ;
                54        01 90 0024A 40$:   MOVB    #1, INCLUDE_STATUS                    ; 1151
                50        69 D0 0024D 41$:   MOVL    OPTION_FLAG, R0                       ; 1162
     03         60        04 E0 00250        BBS     #4, (R0), 42$                         ;
                        00F4 31 00254        BRW     64$                                   ;
       00000000G 00       01 D0 00257 42$:   MOVL    #1, EXCLUDE_FLAG                      ; 1165
                0E     FF A5 E8 0025E        BLBS    DEV_TYPE_ENTRY_STS, 43$               ; 1167
            0040 8F       66 B1 00262        CMPW    EMB+4, #64                            ; 1168
                          07 13 00267        BEQL    43$                                   ;
            0041 8F       66 B1 00269        CMPW    EMB+4, #65                            ; 1169
                          34 12 0026E        BNEQ    47$                                   ;
                50        6A D0 00270 43$:   MOVL    EXCLUDE_MASK, R0                      ; 1172
     13         60        14 E1 00273        BBC     #20, (R0), 45$                        ;
       00000000V 00       00 FB 00277        CALLS   #0, VERIFY_DEVICE                     ; 1175
                06        50 E9 0027E        BLBC    R0, 44$                               ;
              FD A5       01 90 00281        MOVB    #1, DEVICE_STATUS                     ; 1177
                          03 11 00285        BRB     45$                                   ;
                       FD A5 94 00287 44$:   CLRB    DEVICE_STATUS                         ; 1179
                50        6A D0 0028A 45$:   MOVL    EXCLUDE_MASK, R0                      ; 1182
     13         60        15 E1 0028D        BBC     #21, (R0), 47$                        ;
       00000000V 00       00 FB 00291        CALLS   #0, VERIFY_DEVICE_CLASS               ; 1185
                06        50 E9 00298        BLBC    R0, 46$                               ;
              FE A5       01 90 0029B        MOVB    #1, DEV_CLS_STATUS                    ; 1187
                          03 11 0029F        BRB     47$                                   ;
                       FE A5 94 002A1 46$:   CLRB    DEV_CLS_STATUS                        ; 1189
                50        6A D0 002A4 47$:   MOVL    EXCLUDE_MASK, R0                      ; 1193
     11         60        16 E1 002A7        BBC     #22, (R0), 49$                        ;
       00000000V 00       00 FB 002AB        CALLS   #0, VERIFY_ENTRY                      ; 1196
                05        50 E9 002B2        BLBC    R0, 48$                               ;
                65        01 90 002B5        MOVB    #1, ENTRY_STATUS                      ; 1198
                          02 11 002B8        BRB     49$                                   ;
                          65 94 002BA 48$:   CLRB    ENTRY_STATUS                          ; 1200
                50        6A D0 002BC 49$:   MOVL    EXCLUDE_MASK, R0                      ; 1203
     08         60        14 E1 002BF        BBC     #20, (R0), 50$                        ;
                04     FF A5 E9 002C3        BLBC    DEV_TYPE_ENTRY_STS, 50$               ; 1204
                13     FD A5 E8 002C7        BLBS    DEVICE_STATUS, 52$                    ;
     08         60        15 E1 002CB 50$:   BBC     #21, (R0), 51$                        ; 1206
                04     FF A5 E9 002CF        BLBC    DEV_TYPE_ENTRY_STS, 51$               ; 1207
                07     FE A5 E8 002D3        BLBS    DEV_CLS_STATUS, 52$                   ;
     07         60        16 E1 002D7 51$:   BBC     #22, (R0), 53$                        ; 1209
                04        65 E9 002DB        BLBC    ENTRY_STATUS, 53$                     ;
                          53 94 002DE 52$:   CLRB    EXCLUDE_STATUS                        ; 1211
```

```
                                          03  11 002E0           BRB     54$                                         ;
                          53              01  90 002E2 53$:      MOVB    #1, EXCLUDE_STATUS                          ; 1213
              2F          60              14  E1 002E5 54$:      BBC     #20, (R0), 59$                              ; 1216
              2B          60              16  E1 002E9           BBC     #22, (R0), 59$                              ; 1217
                          53              01  90 002ED           MOVB    #1, EXCLUDE_STATUS                          ; 1220
                          11         FC   A5  E9 002F0           BLBC    DEV_SELECTION_REQUIRED, 56$                 ; 1222
                          04              65  E9 002F4           BLBC    ENTRY_STATUS, 55$                           ; 1225
                          1B         FD   A5  E8 002F7           BLBS    DEVICE_STATUS, 58$                          ;
                          19         FF   A5  E9 002FB 55$:      BLBC    DEV_TYPE_ENTRY_STS, 59$                     ; 1226
                          15         FD   A5  E9 002FF           BLBC    DEVICE_STATUS, 59$                          ;
                                          11  11 00303           BRB     58$                                         ; 1228
                          51         FF   A5  9A 00305 56$:      MOVZBL  DEV_TYPE_ENTRY_STS, R1                      ; 1232
                          07              51  E9 00309           BLBC    R1, 57$                                     ;
                          06         FD   A5  E8 0030C           BLBS    DEVICE_STATUS, 58$                          ;
                          05              51  E8 00310           BLBS    R1, 59$                                     ; 1239
                          02              65  E9 00313 57$:      BLBC    ENTRY_STATUS, 59$                           ;
                                          53  94 00316 58$:      CLRB    EXCLUDE_STATUS                              ; 1241
              2F          60              15  E1 00318 59$:      BBC     #21, (R0), 64$                              ; 1246
              2B          60              16  E1 0031C           BBC     #22, (R0), 64$                              ; 1247
                          53              01  90 00320           MOVB    #1, EXCLUDE_STATUS                          ; 1250
                          11         FC   A5  E9 00323           BLBC    DEV_SELECTION_REQUIRED, 61$                 ; 1252
                          04              65  E9 00327           BLBC    ENTRY_STATUS, 60$                           ; 1255
                          1B         FE   A5  E8 0032A           BLBS    DEV_CLS_STATUS, 63$                         ;
                          19         FF   A5  E9 0032E 60$:      BLBC    DEV_TYPE_ENTRY_STS, 64$                     ; 1256
                          15         FE   A5  E9 00332           BLBC    DEV_CLS_STATUS, 64$                         ;
                                          11  11 00336           BRB     63$                                         ; 1258
                          50         FF   A5  9A 00338 61$:      MOVZBL  DEV_TYPE_ENTRY_STS, R0                      ; 1262
                          07              50  E9 0033C           BLBC    R0, 62$                                     ;
                          06         FE   A5  E8 0033F           BLBS    DEV_CLS_STATUS, 63$                         ;
                          05              50  E8 00343           BLBS    R0, 64$                                     ; 1269
                          02              65  E9 00346 62$:      BLBC    ENTRY_STATUS, 64$                           ;
                                          53  94 00349 63$:      CLRB    EXCLUDE_STATUS                              ; 1271
                          32              54  E9 0034B 64$:      BLBC    INCLUDE_STATUS, 67$                         ; 1285
                          2F              53  E9 0034E           BLBC    EXCLUDE_STATUS, 67$                         ;
                          50              6B  D0 00351           MOVL    PARSER_DATA, R0                             ; 1286
                          02              60  91 00354           CMPB    (R0), #2                                    ;
                                          27  12 00357           BNEQ    67$                                         ;
                          50              66  3C 00359           MOVZWL  EMB+4, R0                                   ; 1292
                  0063    8F              50  B1 0035C           CMPW    R0, #99                                     ;
                                          07  13 00361           BEQL    65$                                         ;
                  0064    8F              50  B1 00363           CMPW    R0, #100                                    ; 1293
                                          0C  12 00368           BNEQ    66$                                         ;
                                     F8   A5  9F 0036A 65$:      PUSHAB  LSTLUN                                      ; 1299
              00000000G   00              01  FB 0036D           CALLS   #1, INTERVENE_INCREMENT                     ;
                                          0A  11 00374           BRB     67$                                         ;
                                     F8   A5  9F 00376 66$:      PUSHAB  LSTLUN                                      ; 1305
              00000000G   00              01  FB 00379           CALLS   #1, INTERVENE_OUTPUT                        ;
                          06  00000000G   00  E8 00380 67$:      BLBS    UNKNOWN_ENTRY, 68$                          ; 1313
                          07              54  E9 00387           BLBC    INCLUDE_STATUS, 69$                         ; 1320
                          04              53  E9 0038A           BLBC    EXCLUDE_STATUS, 69$                         ; 1321
                          50              01  D0 0038D 68$:      MOVL    #1, R0                                      ; 1333
                                          04  00390              RET                                                 ;
                                          50  D4 00391 69$:      CLRL    R0                                          ; 1335
                                          04  00393              RET                                                 ;

; Routine Size:  916 bytes,    Routine Base:  $CODE + 0000
;  769          1336  1
;  770          1337  1
```

```
1 ROUTINE VERIFY_ENTRY =
2 Begin
2
2 !++
2 !
2 ! Functional Description:
2 !
2 !       This routine will determine if the current entry matches
2 !       any of the selected entry types. It return TRUE if the
2 !       current entry matches or return FALSE if the current entry
2 !       does NOT match.
2 !
2 ! Calling sequence:
2 !
2 !       VERIFY_ENTRY ()
2 !
2 ! Input parameters:
2 !
2 !       None
2 !
2 ! Output parameters:
2 !
2 !       None
2 !
2 !--
2
2
2 !
2 ! Initialize a status indicator.
2 !
2 Dev_selection_required = false ;
2
2 !
2 ! Determine if device attention entries are selected.
2 !
3 If ((.exclude_mask[exc$v_dev_attentions]) OR
3     (.include_mask[inc$v_dev_attentions]))
2 Then
2     !
2     ! Determine if this entry is for a device attention.
2     !
3     Begin
3     Dev_selection_required = true ;
3     If .emb[emb$w_hd_entry] EQLU EMB$K_DA
3     Then
3         !
3         ! Indicate that this entry does match a selected entry
3         ! type, by returning to the calling routine with a
3         ! true value.
3         !
3         Return true ;
2     End ;
2
2 !
2 ! Determine if bugcheck entries are selected.
2 !
3 If ((.exclude_mask[exc$v_bugchks]) OR
3        (.include_mask[inc$v_bugchks]))
2  Then
2        !
2        ! Determine if this entry is for a bugcheck.
2        !
3        Begin
3        Incr I from 0 to 2 do
4            Begin
4            If .emb[emb$w_hd_entry] EQLU .bugchks[.I]
4            Then
4                !
4                ! Indicate that this entry does match a selected
4                ! entry type, by returning to the calling routine
4                ! with a true value.
4                !
4                Return true ;
3            End ;
2        End ;
2
2  !
2  ! Determine if 'control entries' are selected.
2  !
3  If ((.exclude_mask[exc$v_control_entry]) OR
3        (.include_mask[inc$v_control_entry]))
2  Then
2        !
2        ! Determine if this entry is a 'control entry'.
2        !
3        Begin
3        Incr I from 0 to 6 do
4            Begin
4            If .emb[emb$w_hd_entry] EQLU .control[.I]
4            Tnen
4                !
4                ! Indicate that this entry does match a selected
4                ! entry type, by returning to the calling routine
4                ! with a true value.
4                !
4                Return true ;
3            End ;
2        End ;
2
2  !
2  ! Determine if 'cpu entries' are selected.
2  !
3  If ((.exclude_mask[exc$v_cpu_entry]) OR
3        (.include_mask[inc$v_cpu_entry]))
2  Then
2        !
2        ! Determine if this entry is a 'cpu entry'.
2        !
3        Begin
3        Incr I from 0 to 7 do
4            Begin
4            If .emb[emb$w_hd_entry] EQLU .cpu[.I]
4            Then
4                !
```

```
                ! Indicate that this entry does match a selected
                ! entry type, by returning to the calling routine
                ! with a true value.
                !
                Return true ;
            End ;
        End ;

    !
    ! Determine if device errors are selected.
    !
    If ((.exclude_mask[exc$v_dev_errors]) OR
        (.include_mask[inc$v_dev_errors]))
    Then
        !
        ! Determine if this entry is a device error.
        !
        Begin
        Dev_selection_required = true ;

        Incr I from 0 to 2 do
            Begin
            If .emb[emb$w_hd_entry] EQLU .dev_errors[.I]
            Then
                !
                ! Indicate that this entry does match a selected
                ! entry type, by returning to the calling routine
                ! with a true value.
                !
                Return true ;
            End ;
        End ;

    !
    ! Determine if machine checks are selected.
    !
    If ((.exclude_mask[exc$v_machine_chks]) OR
        (.include_mask[inc$v_machine_chks]))
    Then
        !
        ! Determine if this entry is a machine check.
        !
        Begin
        If .emb[emb$w_hd_entry] EQLU EMB$K_MC
        Then
            !
            ! Indicate that this entry does match a selected
            ! entry type, by returning to the calling routine
            ! with a true value.
            !
            Return true ;
        End ;

    !
    ! Determine if memory entries are selected.
    !
    If ((.exclude_mask[exc$v_memory]) OR
3         (.include_mask[inc$v_memory]))
2     Then
2         !
2         ! Determine if this entry is a 'memory entry'.
2         !
3         Begin
3         Incr I from 0 to 1 do
4             Begin
4             If .emb[emb$w_hd_entry] EQLU .memorys[.I]
4             Then
4                 !
4                 ! Indicate that this entry does match a selected
4                 ! entry type, by returning to the calling routine
4                 ! with a true value.
4                 !
4                 Return true ;
3             End ;
2         End ;
2
2     !
2     ! Determine if device timeouts are selected.
2     !
3     If ((.exclude_mask[exc$v_dev_timeouts]) OR
3         (.include_mask[inc$v_dev_timeouts]))
2     Then
2         !
2         ! Determine if this entry is a device timeouts.
2         !
3         Begin
3         Dev_selection_required = true ;
3
3         If .emb[emb$w_hd_entry] EQLU EMB$K_DT
3         Then
3             !
3             ! Indicate that this entry does match a selected
3             ! entry type, by returning to the calling routine
3             ! with a true value.
3             !
3             Return true ;
2         End ;
2
2
2     !
2     ! Determine if unknown entries have been selected.
2     ! If unknown entries have not been excluded, then see if this is an
2     ! unknown entry. If it is set UNKNOWN_ENTRY true.
2     !
2     ! Initialize the unknown entry indicator (not an unknown entry).
2     !
3     If ((.exclude_mask[exc$v_unknown_entry]) OR
3         (.include_mask[inc$v_unknown_entry]))
2     Then
2         !
2         ! Determine if this is an unknown entry.
2         !
3         Begin
3         If .unknown_entry
```

```
3     Then  Return true ;
2     End ;
2
2 !
2 ! Determine if unsolicited mscp entries are selected.
2 !
3 If ((.exclude_mask[exc$v_unsol_mscp]) OR
3     (.include_mask[inc$v_unsol_mscp]))
2 Then
2     !
2     ! Determine if this entry is an unsolicited mscp entry.
2     !
3     Begin
3     If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP
3     Then
3         !
3         ! Indicate that this entry does match a selected
3         ! entry type, by returning to the calling routine
3         ! with a true value.
3         !
3         Return true ;
2     End ;
2
2 !
2 ! Determine if volume changes are to be excluded.
2 !
4 If ((.exclude_mask[exc$v_volume])
3     OR (.include_mask[inc$v_volume]))
2 Then
2     !
2     ! Determine if this entry is a volume entry.
2     !
3     Begin
3     Dev_selection_required = true ;
3
3     Incr I from 0 to 1 do
4         Begin
4         If .emb[emb$w_hd_entry] EQLU .volume[.I]
4         Then
4             !
4             ! Indicate that this entry does match a selected
4             ! entry type, by returning to the calling routine
4             ! with a true value.
4             !
4             Return true ;
3         End ;
2     End ;
2
2 !
2 ! Indicate that this entry does not match any of the selected
2 ! entry types, by returning to the calling routine with a
2 ! false value.
2 !
2 Return false ;
1 End ;   ! Routine
```

```
                                             003C 00000 VERIFY_ENTRY:
                                                               .WORD   Save R2,R3,R4,R5                    ; 1338
                    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              ; 1368
                    51 00000000G    00   D0 00019              MOVL    EXCLUDE_MASK, R1                    ; 1373
       07           61              09   E0 00020              BBS     #9, (R1), 1$                        ;
                    50              63   D0 00024              MOVL    INCLUDE_MASK, R0                    ; 1374
       0A           60              09   E1 00027              BBC     #9, (R0), 2$                        ;
                    64              01   90 0002B 1$:          MOVB    #1, DEV_SELECTION_REQUIRED          ; 1380
              0062  8F              65   B1 0002E              CMPW    EMB+4, #98                          ; 1381
                                    7D   13 00033              BEQL    16$                                 ;
       07           61              0A   E0 00035 2$:          BBS     #10, (R1), 3$                       ; 1394
                    50              63   D0 00039              MOVL    INCLUDE_MASK, R0                    ; 1395
       10           60              0A   E1 0003C              BBC     #10, (R0), 5$                       ;
                                    50   D4 00040 3$:          CLRL    I                                   ; 1403
                    52       0C A440     9A 00042 4$:          MOVZBL  BUGCHKS[I], R2                      ;
                    65              52   B1 00047              CMPW    R2, EMB+4                           ;
                                    7D   13 0004A              BEQL    20$                                 ;
       F2           50              02   F3 0004C              AOBLEQ  #2, I, 4$                           ; 1401
       07           61              0B   E0 00050 5$:          BBS     #11, (R1), 6$                       ; 1417
                    50              63   D0 00054              MOVL    INCLUDE_MASK, R0                    ; 1418
       10           60              0B   E1 00057              BBC     #11, (R0), 8$                       ;
                                    50   D4 0005B 6$:          CLRL    I                                   ; 1426
                    52       10 A440     9A 0005D 7$:          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$                           ; 1424
       07           61              0C   E0 0006B 8$:          BBS     #12, (R1), 9$                       ; 1440
                    50              63   D0 0006F              MOVL    INCLUDE_MASK, R0                    ; 1441
       10           60              0C   E1 00072              BBC     #12, (R0), 11$                      ;
                                    50   D4 00076 9$:          CLRL    I                                   ; 1449
                    52       18 A440     9A 00078 10$:         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$                          ; 1447
       07           61              0D   E0 00086 11$:         BBS     #13, (R1), 12$                      ; 1463
                    50              63   D0 0008A              MOVL    INCLUDE_MASK, R0                    ; 1464
       13           60              0D   E1 0008D              BBC     #13, (R0), 14$                      ;
                    64              01   90 00091 12$:         MOVB    #1, DEV_SELECTION_REQUIRED          ; 1470
                                    50   D4 00094              CLRL    I                                   ; 1474
                    52       20 A440     9A 00096 13$:         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$                          ; 1472
       07           61              0E   E0 000A4 14$:         BBS     #14, (R1), 15$                      ; 1488
                    50              63   D0 000A8              MOVL    INCLUDE_MASK, R0                    ; 1489
       05           60              0E   E1 000AB              BBC     #14, (R0), 17$                      ;
                    02              65   B1 000AF 15$:         CMPW    EMB+4, #2                           ; 1495
                                    6E   13 000B2 16$:         BEQL    32$                                 ;
                                    61   B5 000B4 17$:         TSTW    (R1)                                ; 1508
                                    07   19 000B6              BLSS    18$                                 ;
                    50              63   D0 000B8              MOVL    INCLUDE_MASK, R0                    ; 1509
```

```
                                   60  B5 000BB          TSTW    (R0)                                    ;
                                   10  18 000BD          BGEQ    21$                                     ;
                                   50  D4 000BF  18$:    CLRL    I                                       ; 1517
                  52        24 A440  9A 000C1  19$:    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$                              ; 1515
                  07        02     A1  E8 000CF  21$:    BLBS    2(R1), 22$                              ; 1531
                  50               63  D0 000D3          MOVL    INCLUDE_MASK, R0                        ; 1532
                  0A        02     A0  E9 000D6          BLBC    2(R0), 24$                              ;
                  64               01  90 000DA  22$:    MOVB    #1, DEV_SELECTION_REQUIRED              ; 1538
            0060  8F               65  B1 000DD          CMPW    EMB+4, #96                              ; 1540
                                   3E  13 000E2  23$:    BEQL    32$                                     ;
      07          61               13  E0 000E4  24$:    BBS     #19, (R1), 25$                          ; 1558
                  50               63  D0 000E8          MOVL    INCLUDE_MASK, R0                        ; 1559
      07          60               13  E1 000EB          BBC     #19, (R0), 26$                          ;
                  2C  00000000G    00  E8 000EF  25$:    BLBS    UNKNOWN_ENTRY, 32$                      ; 1565
      07          61               11  E0 000F6  26$:    BBS     #17, (R1), 27$                          ; 1572
                  50               63  D0 000FA          MOVL    INCLUDE_MASK, R0                        ; 1573
      07          60               11  E1 000FD          BBC     #17, (R0), 29$                          ;
            0065  8F               65  B1 00101  27$:    CMPW    EMB+4, #101                             ; 1579
                                   1A  13 00106  28$:    BEQL    32$                                     ;
      07          61               12  E0 00108  29$:    BBS     #18, (R1), 30$                          ; 1592
                  50               63  D0 0010C          MOVL    INCLUDE_MASK, R0                        ; 1593
      17          60               12  E1 0010F          BBC     #18, (R0), 34$                          ;
                  64               01  90 00113  30$:    MOVB    #1, DEV_SELECTION_REQUIRED              ; 1599
                                   50  D4 00116          CLRL    I                                       ; 1603
                  51        28 A440  9A 00118  31$:    MOVZBL  VOLUME[I], R1                           ;
                  65               51  B1 0011D          CMPW    R1, EMB+4                               ;
                                   04  12 00120          BNEQ    33$                                     ;
                  50               01  D0 00122  32$:    MOVL    #1, R0                                  ; 1610
                                       04 00125          RET                                             ;
      EE          50               01  F3 00126  33$:    AOBLEQ  #1, I, 31$                              ; 1601
                                   50  D4 0012A  34$:    CLRL    R0                                      ; 1619
                                       04 0012C          RET                                             ; 1620

; Routine Size:  301 bytes,    Routine Base:  $CODE + 0394


; 1055          1621  1
```

```
1 GLOBAL ROUTINE DEVICE_TYPE_ENTRY =
2 Begin
2
2 !++
2 !
2 ! Functional Description:
2 !
2 !       This routine will determine if the current entry is a device
2 !       type entry; (device attention, device error, device timeout,
2 !       volume dismount, volume mount). It return TRUE if the current
2 !       entry matches any of the device type entries or return FALSE
2 !       if the current entry does NOT match.
2 !
2 ! Calling sequence:
2 !
2 !       DEVICE_ENTRY_TYPE ()
2 !
2 ! Input parameters:
2 !
2 !       None
2 !
2 ! Output parameters:
2 !
2 !       None
2 !
2 !--
2
2 OWN
2     Device_entries:     VECTOR [6,byte,unsigned] ! Storage for device type
2                                                  ! entries.
2                         Initial (BYTE
2                                 (EMB$K_DA,       ! Device attentions
2                                 EMB$K_DE,        ! Device errors
2                                 EMB$K_DT,        ! Device timeouts
2                                 EMB$K_LM,
2                                 EMB$K_SP,        ! Log message
2                                 EMB$K_LOGMSCP)) ;! Unsolicited mscp msg
2
2 !
2 ! Determine if the current entry is a device type entry.
2 !
2 Incr I from 0 to 5 do
3     Begin
3     If .emb[emb$w_hd_entry] EQLU .device_entries[.I]
3     Then
3         !
3         ! Indicate that this is a device type entry, by
3         ! returning to the calling routine with a true value.
3
3         Return true ;
2     End ;
2
2 !
2 ! Indicate that this is NOT a device type entry, by returning
2 ! to the calling routine with a false value.
2 !
2 Return false ;
```

```
; 1114          1679  2
; 1115          1680  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                ; 1622
                                               50  D4 00002       CLRL    I                                              ; 1665
                            51 00000000'0040  9A 00004 1$:        MOVZBL  DEVICE_ENTRIES[I], R1                          ;
              00000000G  00                 51  B1 0000C          CMPW    R1, EMB+4                                      ;
                                            04  12 00013          BNEQ    2$                                             ;
                         50                 01  D0 00015          MOVL    #1, R0                                         ; 1671
                                                04 00018          RET                                                    ;
           E7            50                 05  F3 00019 2$:      AOBLEQ  #5, I, 1$                                      ; 1663
                                            50  D4 0001D          CLRL    R0                                             ; 1678
                                                04 0001F          RET                                                    ; 1680

; Routine Size:  32 bytes,    Routine Base:  $CODE + 04C1


; 1116          1681  1
```

```
1 ROUTINE VERIFY_DEVICE_CLASS =
2 Begin
2
2 !++
2 !
2 ! Functional Descri  ion:
2 !
2 !       This routine will determine if the device recorded by the
2 !       current entry matches any of the selected device class(es).
2 !       It return TRUE if the current entry matches or return FALSE
2 !       if the current entry does NOT match.
2 !
2 ! Calling sequence:
2 !
2 !       VERIFY_DEVICE_CLASS ()
2 !
2 ! Input parameters:
2 !
2 !       None
2 !
2 ! Output parameters:
2 !
2 !       None
2 !
2 !--
2
2 !
2 ! Determine whether this is a unsolicited mscp entry and
2 ! whether to continue.
2 !
2 If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
2    NOT .include_mask[inc$v_disks] AND
2     NOT .include_mask[inc$v_tapes]
2 Then
2    Return false ;
2
2 !
2 ! Determine if 'BUS' entries are selected.
2 !
3 If ((.exclude_mask[exc$v_buses]) OR
3    (.include_mask[inc$v_buses]))
2 Then
2    !
2    ! Determine if the device recorded by this entry, matches the
2    ! selected device class.
2    !
3    Begin
5    If  ((.emb[emb$w_hd_entry] EQLU EMB$K_LM AND
3        .emb[emb$b_lm_class] EQLU DC$_BUS) OR
3
5        ((.emb[emb$w_hd_entry] EQLU EMB$K_SP AND
3        .emb[emb$b_sp_class] EQLU DC$_BUS) OR
3
4        (.emb[emb$b_dv_class] EQLU DC$_BUS)
3    Then
3        !
3        ! Indicate that this entry does match a selected device
```

```
3         ! class, by returning to the calling routine with a
3         ! true value.
3         !
3         Return true ;
2       End ;
2
2 !
2 ! Determine if 'DISK' entries are selected.
2 !
3 If ((.exclude_mask[exc$v_disks]) OR
3     (.include_mask[inc$v_disks]))
2 Then
2       !
2       ! Determine if the device recorded uy this entry, matches the
2       ! selected device class.
2       !
3       Begin
4       If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
4           (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
3       Then
3         !
3         ! Determine if the device recorded by this volume
3         ! mount or dismount is a 'disk' type device.
3         !
4         Begin
4         If NOT TRANSLATE_CLASS (emb[emb$t_vm_namtxt],DC$_DISK)
4         Then
4             !
4             ! Indicate that the device recorded by this entry is
4             ! not a 'disk', by returning to the calling routine
4             ! with a false value.
4             !
4             Return false
4         Else
4             Return true ;
3         End ;
3
5       If ( ((.emb[emb$w_hd_entry] EQLU EMB$K_LM) AND
4               (.emb[emb$b_lm_class] EQLU DC$_DISK)) OR
4
5           ((.emb[emb$w_hd_entry] EQLU EMB$K_SP) AND
4               (.emb[emb$b_sp_class] EQLU DC$_DISK)) OR
4
4       ! Entry type must be either a device error, timeout, or attention.
4       !
4           (.emb[emb$b_dv_class] EQLU DC$_DISK) )
3       Then
3         !
3         ! Indicate that this entry does match a selected
3         ! device class, by returning to the calling routine
3         ! with a true value.
3         !
3         Return true ;
3
3       !
3       ! Determine whether this is disk related unsolicited mscp entry.
3       !
```

```
3      If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
3         CH$EQL (2,emb[driver_type],2,CH$PTR(uplit('DISK')))
3      Then
3          ! Yes, return to the calling routine with a true value.
3          !
3          Return true ;
2      End ;
2
2  !
2  ! Determine if 'REALTIME' entries are selected.
2  !
3  If ((.exclude_mask[exc$v_realtime]) OR
3      (.include_mask[inc$v_realtime]))
2  Then
2      !
2      ! Determine if the device recorded by this entry, matches the
2      ! selected device class.
2      !
3      Begin
3      If .emb[emb$b_dv_class] EQLU DC$_REALTIME
3      Then
3          !
3          ! Indicate that this entry does match a selected
3          ! device class, by returning to the calling routine
3          ! with a true value.
3          !
3          Return true ;
2      End ;
2
2  !
2  ! Determine if 'SYNCHRONOUS COMMUNICATION' entries are selected.
2  !
3  If ((.exclude_mask[exc$v_sync_comm]) OR
3      (.include_mask[inc$v_sync_comm]))
2  Then
2      !
2      ! Determine if the device recorded by this entry, matches the
2      ! selected device class.
2      !
3      Begin
3      If .emb[emb$b_dv_class] EQLU DC$_SCOM
3      Then
3          !
3          ! Indicate that this entry does match a selected
3          ! device class, by returning to the calling routine
3          ! with a true value.
3          !
3          Return true ;
2      End ;
2
2  !
2  ! Determine if 'TAPE' entries are selected.
2  !
3  If ((.exclude_mask[exc$v_tapes]) OR
3      (.include_mask[inc$v_tapes]))
2  Then
2      !
```

```
2       ! Determine if the device recorded by this entry, matches the
2       ! selected device class.
2       !
3       Begin
4       If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
4           (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
3       Then
3           !
3           ! Determine if the device recorded by this volume
3           ! mount or dismount is a 'tape' type device.
3           !
4           Begin
4           If NOT TRANSLATE_CLASS (emb[emb$t_vm_namtxt],DC$_TAPE)
4           Then
4               !
4               ! Indicate that the device recorded by this entry is
4               ! not a 'tape', by returning to the calling routine
4               ! with a false value.
4               !
4               Return false
4           Else
4               Return true ;
3           End ;
3
5       If ( ((.emb[emb$w_hd_entry] EQLU EMB$K_LM) AND
4               (.emb[emb$b_lm_class] EQLU DC$_TAPE)) OR
4
5           ((.emb[emb$w_hd_entry] EQLU EMB$K_SP) AND
4               (.emb[emb$b_sp_class] EQLU DC$_TAPE)) OR
4
4       ! Entry type must be either a device error, timeout, or attention.
4       !
4           (.emb[emb$b_dv_class] EQLU DC$_TAPE) )
3       Then
3           !
3           ! Indicate that this entry does match a selected
3           ! device class, by returning to the calling routine
3           ! with a true value.
3           !
3           Return true ;
3
3       !
3       ! Determine whether this is tape related unsolicited mscp entry.
3       !
3       if .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP AND
3           CH$EQL (2,emb[driver_type],2,CH$PTR(uplit('TAPE')))
3       Then
3           ! Yes, return to the calling routine with a true value.
3           !
3           Return true ;
2       End ;
2
2   !
2   ! Determine if 'MISC' entries are selected.
2   !
2   !If ((.exclude_mask[exc$v_misc]) OR
2   !    (.include_mask[inc$v_misc]))
```

```
2 !Then
2       !
2       ! Determine if the device recorded by this entry, matches the
2       ! selected device class.
2       !
2 !     Begin
2 !     If .emb[emb$b_dv_class] EQLU DC$_MISC
2 !     Then
2           !
2           ! Indicate that thi^ entry does match a selected
2           ! device class, by  eturning to the calling routine
2           ! with a true value.
2           !
2 !         Return true ;
2 !     End ;
2
2 !
2 ! Determine if 'LP' entries are selected.
2 !
2 !If ((.exclude_mask[exc$v_line_printr]) OR
2 !     (.include_mask[inc$v_line_printr]))
2 !Then
2       !
2       ! Determine if the device recorded by this entry, matches the
2       ! selected device class.
2       !
2 !     Begin
2 !     If .emb[emb$b_dv_class] EQLU DC$_LP
2 !     Then
2           !
2           ! Indicate that this entry does match a selected
2           ! device class, by returning to the calling routine
2           ! with a true value.
2           !
2 !         Return true ;
2 !     End ;
2
2 !
2 ! Determine if 'JOURNAL' entries are selected.
2 !
2 !If ((.exclude_mask[exc$v_journal]) OR
2 !     (.include_mask[inc$v_journal]))
2 !Then
2       !
2       ! Determine if the device recorded by this entry, matches the
2       ! selected device class.
2       !
2 !     Begin
2 !     If .emb[emb$b_dv_class] EQLU DC$_JOURNAL
2 !     Then
2           !
2           ! Indicate that this entry does match a selected
2           ! device class, by returning to the calling routine
2           ! with a true value.
2           !
2 !         Return true ;
2 !     End ;
```

```
; 1403           1967  2
; 1404           1968  2 !
; 1405           1969  2 ! Indicate that this entry does not match any of the selected
; 1406           1970  2 ! device classes, by returning to the calling routine with a
; 1407           1971  2 ! false value.
; 1408           1972  2 !
; 1409           1973  2 Return false ;
; 1410           1974  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                      ; 1682
                         55 00000000G  00  9E 00002               MOVAB   EXCLUDE_MASK, R5                      ;
                         54 00000000G  00  9E 00009               MOVAB   INCLUDE_MASK, R4                      ;
                         53 00000000G  00  9E 00010               MOVAB   EMB+16, R3                            ;
                         52        F4  A3  3C 00017               MOVZWL  EMB+4, R2                             ; 1712
               0065      8F            52  B1 0001B               CMPW    R2, #101                              ;
                                       11  12 00020               BNEQ    1$                                    ;
                         50            64  D0 00022               MOVL    INCLUDE_MASK, R0                      ; 1713
          0A             60            02  E0 00025               BBS     #2, (R0), 1$                          ;
                         50            64  D0 00029               MOVL    INCLUDE_MASK, R0                      ; 1714
                         03        01  A0  E8 0002C               BLBS    1(R0), 1$                             ;
                                     010A  31 00030               BRW     28$                                   ;
                         51            65  D0 00033 1$:           MOVL    EXCLUDE_MASK, R1                      ; 1721
          07             61            01  E0 00036               BBS     #1, (R1), 2$                          ;
                         50            64  D0 0003A               MOVL    INCLUDE_MASK, R0                      ; 1722
          21             60            01  E1 0003D               BBC     #1, (R0), 5$                          ;
               0064      8F            52  B1 00041 2$:           CMPW    R2, #100                              ; 1729
                                       06  12 00046               BNEQ    3$                                    ;
                 80      8F            63  91 00048               CMPB    EMB+16, #128                          ; 1730
                                       77  13 0004C               BEQL    13$                                   ;
               0063      8F            52  B1 0004E 3$:           CMPW    R2, #99                               ; 1732
                                       06  12 00053               BNEQ    4$                                    ;
                 80      8F            63  91 00055               CMPB    EMB+16, #128                          ; 1733
                                       7B  13 00059               BEQL    16$                                   ;
                 80      8F        0C  A3  91 0005B 4$:           CMPB    EMB+28, #128                          ; 1735
                                       74  13 00060               BEQL    16$                                   ;
          07             61            02  E0 00062 5$:           BBS     #2, (R1), 6$                          ; 1748
                         50            64  D0 00066               MOVL    INCLUDE_MASK, R0                      ; 1749
          45             60            02  E1 00069               BBC     #2, (R0), 11$                         ;
               0040      8F            52  B1 0006D 6$:           CMPW    R2, #64                               ; 1756
                                       07  13 00072               BEQL    7$                                    ;
               0041      8F            52  B1 00074               CMPW    R2, #65                               ; 1757
                                       04  12 00079               BNEQ    8$                                    ;
                                       01  DD 0007B 7$:           PUSHL   #1                                    ; 1764
                                       78  11 0007D               BRB     20$                                   ;
                         50        F4  A3  3C 0007F 8$:           MOVZWL  EMB+4, R0                             ; 1776
```

```
           0064   8F              50  B1 00083          CMPW    R0, #100                             :
                                  05  12 00088          BNEQ    9$                                   :
                  01              63  91 0008A          CMPB    EMB+16, #1                           : 1777
                                  47  13 0008D          BEQL    16$                                  :
           0063   8F              50  B1 0008F 9$:      CMPW    R0, #99                              : 1779
                                  05  12 00094          BNEQ    10$                                  :
                  01              63  91 00096          CMPB    EMB+16, #1                           : 1780
                                  79  13 00099          BEQL    22$                                  :
                  01        0C    A3  91 0009B 10$:     CMPB    EMB+28, #1                           : 1784
                                  7F  13 0009F          BEQL    24$                                  :
           0065   8F              50  B1 000A1          CMPW    R0, #101                             : 1796
                                  0A  12 000A6          BNEQ    11$                                  :
       00000000'  00        02    A3  B1 000A8          CMPW    EMB+18, P.AAA                        : 1797
                                  74  13 000B0          BEQL    26$                                  :
                  51              65  D0 000B2 11$:     MOVL    EXCLUDE_MASK, R1                     : 1807
    07            61              06  E0 000B5          BBS     #6, (R1), 12$                        :
                  50              64  D0 000B9          MOVL    INCLUDE_MASK, R0                     : 1808
    07            60              06  E1 000BC          BBC     #6, (R0), 14$                        :
             60   8F        0C    A3  91 000C0 12$:     CMPB    EMB+28, #96                          : 1815
                                  72  13 000C5 13$:     BEQL    27$                                  :
                                  61  95 000C7 14$:     TSTB    (R1)                                 : 1828
                                  07  19 000C9          BLSS    15$                                  :
                  50              64  D0 000CB          MOVL    INCLUDE_MASK, R0                     : 1829
                                  60  95 000CE          TSTB    (R0)                                 :
                                  06  18 000D0          BGEQ    17$                                  :
                  20        0C    A3  91 000D2 15$:     CMPB    EMB+28, #32                          : 1836
                                  61  13 000D6 16$:     BEQL    27$                                  :
                  07        01    A1  E8 000D8 17$:     BLBS    1(R1), 18$                           : 1849
                  50              64  D0 000DC          MOVL    INCLUDE_MASK, R0                     : 1850
                  5A        01    A0  E9 000DF          BLBC    1(R0), 28$                           :
                  50        F4    A3  3C 000E3 18$:     MOVZWL  EMB+4, R0                            : 1857
           0040   8F              50  B1 000E7          CMPW    R0, #64                              :
                                  07  13 000EC          BEQL    19$                                  :
           0041   8F              50  B1 000EE          CMPW    R0, #65                              : 1858
                                  11  12 000F3          BNEQ    21$                                  :
                                  02  DD 000F5 19$:     PUSHL   #2                                   : 1865
                            0F    A3  9F 000F7 20$:     PUSHAB  EMB+31                               :
       00000000V  00              02  FB 000FA          CALLS   #2, TRANSLATE_CLASS                  :
                  35              50  E8 00101          BLBS    R0, 27$                              :
                                  37  11 00104          BRB     28$                                  : 1874
                  50        F4    A3  3C 00106 21$:     MOVZWL  EMB+4, R0                            : 1877
           0064   8F              50  B1 0010A          CMPW    R0, #100                             :
                                  05  12 0010F          BNEQ    23$                                  :
                  02              63  91 00111          CMPB    EMB+16, #2                           : 1878
                                  23  13 00114 22$:     BEQL    27$                                  :
           0063   8F              50  B1 00116 23$:     CMPW    R0, #99                              : 1880
                                  05  12 0011B          BNEQ    25$                                  :
                  02              63  91 0011D          CMPB    EMB+16, #2                           : 1881
                                  17  13 00120 24$:     BEQL    27$                                  :
                  02        0C    A3  91 00122 25$:     CMPB    EMB+28, #2                           : 1885
                                  11  13 00126 26$:     BEQL    27$                                  :
           0065   8F              50  B1 00128          CMPW    R0, #101                             : 1897
                                  0E  12 0012D          BNEQ    28$                                  :
       00000000'  00        02    A3  B1 0012F          CMPW    EMB+18, P.AAB                        : 1898
                                  04  12 00137          BNEQ    28$                                  :
                  50              01  D0 00139 27$:     MOVL    #1, R0                               : 1902
                                      04 0013C          RET                                          :
```

```
                                      50  D4 0013D 28$:       CLRL    R0                                     ; 1974
                                          04 0013F            RET                                            ;

; Routine Size:  320 bytes,    Routine Base:  $CODE + 04E1

; 1411          1975  1
```

```
1 Routine VERIFY_DEVICE =
2 Begin
2
2 !++
2 !
2 !--
2
2 Local
2     Dev_name,
2     Dev_name_length,
2     Dev_unit,
2     Status ;
2
2 Bind
2     lm_name_length = emb[emb$t_lm_devnam] : BYTE,
2     sp_name_length = emb[emb$t_sp_devnam] : BYTE,
2     dv_name_length = emb[emb$t_dv_name] : BYTE ;
2
2 !
2 ! Determine whether this is an unsolicited mscp entry and
2 ! return with a false value if so (logmscp entries are not
2 ! applicable to a specific device).
2 !
2 If .emb[emb$w_hd_entry] EQLU EMB$K_LOGMSCP
2 Then
2     Return false ;
2
2 !
2 ! Determine the type of entry so that the comparison for the
2 ! device class is made against the appropriate field in the entry.
2 !
2 ! Determine if this a log message entry.
2 !
2 If .emb[emb$w_hd_entry] EQLU EMB$K_LM
2 Then
2     !
2     ! Entry type is a log message, get the device name,
2     ! name length, and unit number.
2     !
3     Begin
3     Dev_name = emb[emb$t_lm_devnam] + 1 ;
3     Dev_name_length = .lm_name_length ;
3     Dev_unit = .emb[emb$w_lm_unit] ;
3     End
2 Else
2     !
2     ! Determine if this is a log status entry.
2     !
3     Begin
3     If .emb[emb$w_hd_entry] EQLU EMB$K_SP
3     Then
3         !
3         ! Entry type is a log status, get the device name,
3         ! name length, and unit number.
3         !
4         Begin
4         Dev_name = emb[emb$t_sp_devnam] + 1 ;
```

```
; 1470      2033 4          Dev_name_length = .sp_name_length ;
; 1471      2034 4          Dev_unit = .emb[emb$w_sp_unit] ;
; 1472      2035 4          End
; 1473      2036 3      Else
; 1474      2037 3          !
; 1475      2038 3          ! Determine if this a volume mount/dismount entry.
; 1476      2039 3          !
; 1477      2040 4          Begin
; 1478      2041 5          If ((.emb[emb$w_hd_entry] EQLU EMB$K_VM) OR
; 1479      2042 5              (.emb[emb$w_hd_entry] EQLU EMB$K_VD))
; 1480      2043 4          Then
; 1481      2044 4              !
; 1482      2045 4              ! Entry type is a either a volume mount/dismount, get
; 1483      2046 4              ! the device name, name length, and unit number.
; 1484      2047 4              !
; 1485      2048 5              Begin
; 1486      2049 5              Dev_name = emb[emb$t_vm_namtxt] ;
; 1487      2050 5              Dev_name_length = .emb[emb$b_vm_namlng] ;
; 1488      2051 5              Dev_unit = .emb[emb$w_vm_unit] ;
; 1489      2052 5              End
; 1490      2053 4          Else
; 1491      2054 4              !
; 1492      2055 4              ! Entry type must be either a device error, device timeount,
; 1493      2056 4              ! or a device attention, get the device name, name length, and
; 1494      2057 4              ! unit number.
; 1495      2058 4              !
; 1496      2059 5              Begin
; 1497      2060 5              Dev_name = emb[emb$t_dv_name] + 1 ;
; 1498      2061 5              Dev_name_length = .dv_name_length ;
; 1499      2062 5              Dev_unit = .emb[emb$w_dv_unit] ;
; 1500      2063 4              End ;
; 1501      2064 3          End ;
; 1502      2065 2      End ;
; 1503      2066 2
; 1504      2067 2 !
; 1505      2068 2 ! Call the search queue routine to determine if the device recorded by
; 1506      2069 2 ! this entry matches any of the selected devices.
; 1507      2070 2 !
; 1508      2071 2 Status = SEARCH_QUEUE (.dev_name,dev_name_length,dev_unit) ;
; 1509      2072 2
; 1510      2073 2 !
; 1511      2074 2 ! Return the status from the search queue operation to the
; 1512      2075 2 ! calling routine.
; 1513      2076 2 !
; 1514      2077 2 .Status
; 1515      2078 1 End ;   ! Routine



                                        0004 00000 VERIFY_DEVICE:
                                                          .WORD    Save R2                          ; 1976
               52 00000000G  00  9E 00002                 MOVAB    EMB+4, R2                        ;
               5E            08  C2 00009                 SUBL2    #8, SP                           ;
               50            62  3C 0000C                 MOVZWL   EMB+4, R0                        ; 1999
          0065 8F            50  B1 0000F                 CMPW     R0, #101                         ;
```

```
                                       03  12 00014          BNEQ    1$                                          ;
                                       50  D4 00016          CLRL    R0                                          ; 2001
                                           04 00018          RET                                                 ;
                  0064    8F           50  B1 00019 1$:      CMPW    R0, #100                                    ; 2009
                                       0F  12 0001E          BNEQ    2$                                          ;
                          51      11   A2  9E 00020          MOVAB   EMB+21, DEV_NAME                            ; 2016
                    04    AE      10   A2  9A 00024          MOVZBL  LM_NAME_LENGTH, DEV_NAME_LENGTH             ; 2017
                          6E      0E   A2  3C 00029          MOVZWL  EMB+18, DEV_UNIT                            ; 2018
                                       3C  11 0002D          BRB     7$                                          ; 2009
                  0063    8F           50  B1 0002F 2$:      CMPW    R0, #99                                     ; 2025
                                       0B  12 00034          BNEQ    3$                                          ;
                          51      3D   A2  9E 00036          MOVAB   EMB+65, DEV_NAME                            ; 2032
                    04    AE      3C   A2  9A 0003A          MOVZBL  SP_NAME_LENGTH, DEV_NAME_LENGTH             ; 2033
                                       26  11 0003F          BRB     6$                                          ; 2034
                  0040    8F           50  B1 00041 3$:      CMPW    R0, #64                                     ; 2041
                                       07  13 00046          BEQL    4$                                          ;
                  0041    8F           50  B1 00048          CMPW    R0, #65                                     ; 2042
                                       0F  12 0004D          BNEQ    5$                                          ;
                          51      1B   A2  9E 0004F 4$:      MOVAB   EMB+31, DEV_NAME                            ; 2049
                    04    AE      1A   A2  9A 00053          MOVZBL  EMB+30, DEV_NAME_LENGTH                     ; 2050
                          6E      18   A2  3C 00058          MOVZWL  EMB+28, DEV_UNIT                            ; 2051
                                       0D  11 0005C          BRB     7$                                          ; 2041
                          51      3B   A2  9E 0005E 5$:      MOVAB   EMB+63, DEV_NAME                            ; 2060
                    04    AE      3A   A2  9A 00062          MOVZBL  DV_NAME_LENGTH, DEV_NAME_LENGTH             ; 2061
                          6E      26   A2  3C 00067 6$:      MOVZWL  EMB+42, DEV_UNIT                            ; 2062
                                       5E  DD 0006B 7$:      PUSHL   SP                                          ; 2071
                                  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                                                 ; 2078

; Routine Size:  122 bytes,    Routine Base:  $CODE + 0621


; 1516          2079  1
```

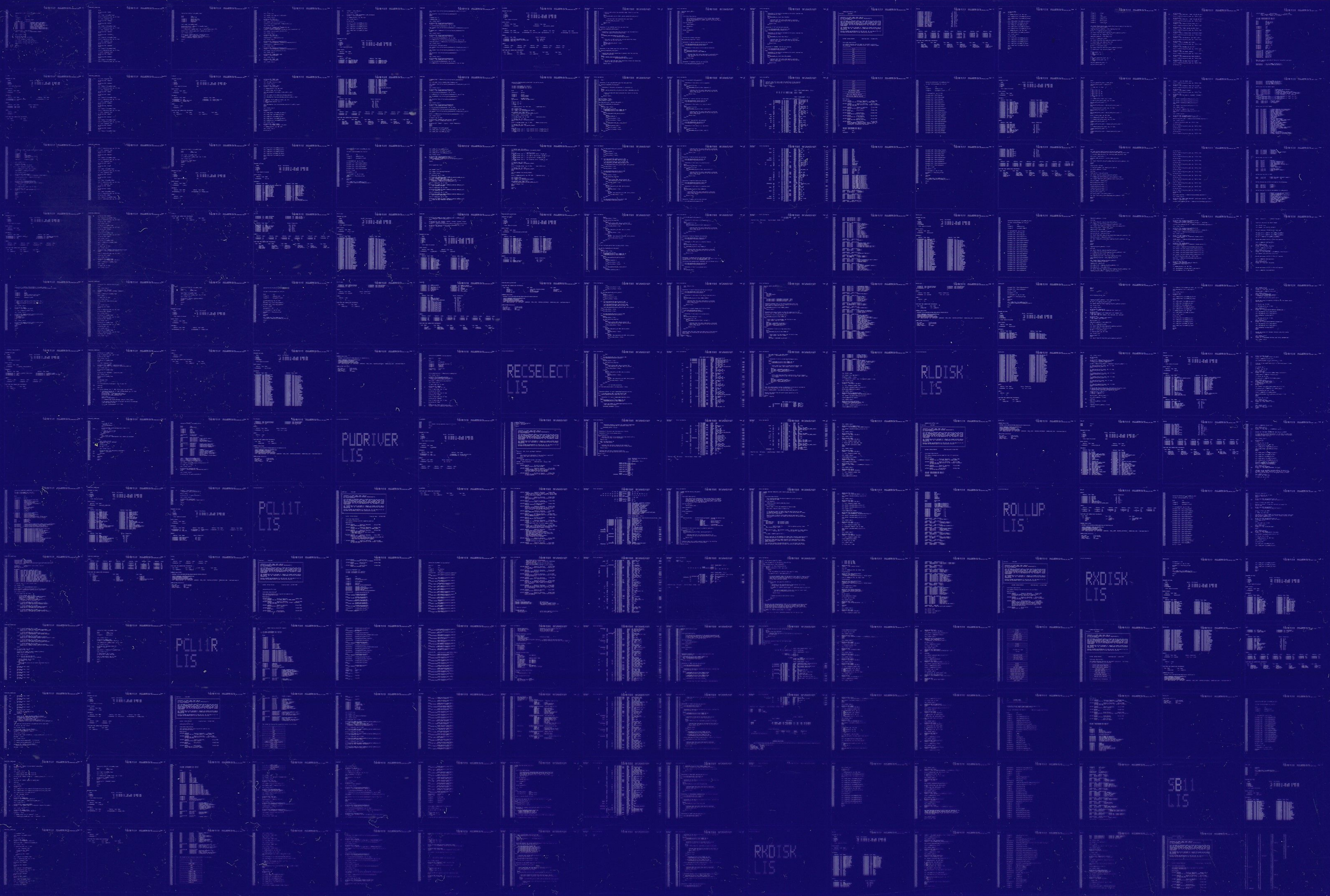
```
1 GLOBAL ROUTINE TRANSLATE_CLASS (search_name,dev_class) =
2 Begin
2
2 !++
2 !
2 ! Functional Description:
2 !
2 !     This routine searches the device tables to verify the device
2 !     class and device name.
2 !
2 ! Calling Sequence:
2 !
2 !     TRANSLATE_CLASS (search_name,dev_class)
2 !
2 ! Input Parameters:
2 !
2 !     Search name = First two characters of device name
2 !
2 !     Dev_class = Device class to search for.
2 !
2 !
2 !     If the device class is found, then the specified device name
2 !     is compared against the device names in the device specific table.
2 !     Returns true if both match.
2 !
2 !     Returns false if device class and/or device name doesn't match.
2 !     (This should eventually be caught and handled by the parse_devname
2 !     routine.)
2 !
2 !--
2
2 EXTERNAL
2     Dev_addrs_ptr:      REF VECTOR [,long],
2     Dev_class_ptr:      REF VECTOR [,word],
2     Max_classes:        REF VECTOR [,byte];
2
2 OWN
2     I:          BYTE Initial (1),        ! Device address pointer index
2     Max_classes_value:  BYTE ;
2
2 LOCAL
2     Dev_specific_tbl:   REF VECTOR [,word], ! Device specific table address
2     K:          Initial (0) ;            ! Device specific table index
2
2 BIND
2     Cs_name = CH$PTR (uplit('CS')) ;
2
2 !
2 ! Device class ptr is the address of a table that contains supported device
2 ! classes and pointers to the device class specific information tables.
2 !
2 ! The device class specific table contains the supported device names,
2 ! image name pointers (image that needs to get activated), and transfer
2 ! address pointers.
2 !
2 ! This routine locates the matching device ( ass retrieves the device
2 ! specific pointer and matches the specified device name against those
```

```
2 ! in the device specific table.
2 !
2 ! Loop through all of the device class entries.
2 !
2 Max_classes_value = max_classes[0] ;
2
2 Incr I from 1 to .max_classes_value do
3     Begin
3     If .dev_class_ptr[.I] EQL .dev_class
3     Then
4         Begin
4         !
4         ! Get the address of a device class specific table.
4         !
4         Dev_specific_tbl = .dev_addrs_ptr[.I] ;
4
4         !
4         ! Initialize another index for the device class specific table so don't
4         ! lose the current position. Determine if the contents of the device
4         ! name field is valid OR whether the end of the device name entries
4         ! in the table has been reached.
4         !
4         K = 1 ;
4         Until (.K EQL .dev_specific_tbl[0]) do
5             Begin
5             !
5             ! Determine if the selected device name matches any of the
5             ! device names recorded in this table.
5             !
5             If CH$EQL (2, CH$PTR(.search_name), 2, CH$PTR(dev_specific_tbl[.K]))
5             Then
5                 !
5                 ! The device names match. Using the class dir table index,
5                 ! get the corresponding device class.
5                 !
5                 Return true ;
5
5             !
5             ! Update the device name pointer indices.
5             !
5             K = .K + 1 ;
4             End ;
3         End ;
2     End ;
2
2
2 !
2 ! The name for the console device 'CSA' is not included in the device name
2 ! tables contained in ERFLIB.TLB. It really is a second device name for
2 ! the RX device which is included in the device tables. There should be
2 ! a table that includes devices like these, however because there is only
2 ! one at this time, it is checked for explicitly.
2 !
2 If CH$EQL (2, CH$PTR(.search_name), 2, cs_name)
2 Then
2     !
2     ! This is a 'CS' entry, determine whether the 'CS' device class
```

```
; 1632          2194  2       ! matches the device class being searched for.
; 1633          2195  2       !
; 1634          2196  3       Begin
; 1635          2197  3       If .dev_class EQL DC$_DISK
; 1636          2198  3       Then
; 1637          2199  3           ! Indicate that the device class matches by returning with
; 1638          2200  3           ! a true value.
; 1639          2201  3           !
; 1640          2202  3           Return true ;
; 1641          2203  2       End ;
; 1642          2204  2
; 1643          2205  2
; 1644          2206  2 !
; 1645          2207  2 ! Could not locate a class for this device name.
; 1646          2208  2 !
; 1647          2209  2 Return false ;
; 1648          2210  2
; 1649          2211  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              ; 2080
                                  55 00000000'  00   9E 00002          MOVAB   MAX_CLASSES_VALUE, R5
                                                52   D4 00009          CLRL    K                                             ; 2081
                                  65 00000000G  00   90 0000B          MOVB    MAX_CLASSES, MAX_CLASSES_VALUE                ; 2141
                                  54            65   9A 00012          MOVZBL  MAX_CLASSES_VALUE, R4                         , 2143
                                                50   D4 00015          CLRL    I                                             ; 2145
                                                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                             ; 2151
                                  53          6140   D0 00032          MOVL    (R1)[I], DEV_SPECIFIC_TBL
                                  52            01   D0 00036          MOVL    #1, K                                         ; 2159
              52            63             10   00   ED 00039 2$:      CMPZV   #0, #16, (DEV_SPECIFIC_TBL), K                ; 2160
                                                0B   13 0003E          BEQL    3$                                            ;
                                6342       04   BC   B1 00040          CMPW    @SEARCH_NAME, (DEV_SPECIFIC_TBL)[K]           ; 2166
                                                18   13 00045          BEQL    4$                                            ;
                                                52   D6 00047          INCL    K                                             ; 2177
                                                EE   11 00049          BRB     2$                                            ; 2160
```

```
                CA          50            54  F3 0004B 3$:     AOBLEQ  R4, I, 1$                          ; 2143
            00000000'       00     04     BC  B1 0004F         CMPW    @SEARCH_NAME, CS_NAME              ; 2190
                                          0A  12 00057         BNEQ    5$                                 ;
                            01     08     AC  D1 00059         CMPL    DEV_CLASS, #1                      ; 2197
                                          04  12 0005D         BNEQ    5$                                 ;
                            50            01  D0 0005F 4$:     MOVL    #1, R0                             ; 2202
                                              04 00062         RET                                        ;
                                          50  D4 00063 5$:     CLRL    R0                                 ; 2209
                                              04 00065         RET                                        ; 2211

; Routine Size:  102 bytes,    Routine Base:  $CODE + 069B


; 1650          2212  1
; 1651          2213  1
; 1652          2214  1 End
; 1653          2215  0 ELUDOM



                                                               .EXTRN  LIB$SIGNAL

;                                 PSECT SUMMARY
;
;       Name                    Bytes                      Attributes
;
;  $OWN$                           56  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  $CODE                         1793  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
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1            18619        72         0      1000         00:01.9




;                                  COMMAND QUALIFIERS

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

; Size:         1793 code + 68 data bytes
; Run Time:        00:40.9
; Elapsed Time:    01:21.8
; Lines/CPU Min:     3251
; Lexemes/CPU-Min: 19926
; Memory Used:  349 pages
; Compilation Complete
```

0153 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY



RECSELECT
LIS

RLDISK
LIS

PUDRIVER
LIS

PCL11T
LIS

ROLLUP
LIS

RXDISK
LIS

PCL11R
LIS

SB11
LIS

RKDISK
LIS