L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
_$

Ps
--

YZ

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$
```

```
**FILE**ID**SYSCHKPRO                             E  5                                                                  SY
                                                                                                                        V0
```

```
  SSSSSSSS  YY      YY    SSSSSSSS    CCCCCCCC  HH      HH  KK      KK  PPPPPPPP    RRRRRRRR      OOOOOO
  SSSSSSSS  YY      YY    SSSSSSSS    CCCCCCCC  HH      HH  KK      KK  PPPPPPPP    RRRRRRRR      OOOOOO
SS          YY      YY  SS          CC          HH      HH  KK      KK  PP      PP  RR      RR  OO      OO
SS          YY      YY  SS          CC          HH      HH  KK      KK  PP      PP  RR      RR  OO      OO
SS            YY  YY    SS          CC          HH      HH  KK    KK    PP      PP  RR      RR  OO      OO
SS            YY  YY    SS          CC          HH      HH  KK    KK    PP      PP  RR      RR  OO      OO
  SSSSSS        YY        SSSSSS    CC          HHHHHHHHHH  KKKKKK      PPPPPPPP    RRRRRRRR    OO      OO
  SSSSSS        YY        SSSSSS    CC          HHHHHHHHHH  KKKKKK      PPPPPPPP    RRRRRRRR    OO      OO
        SS      YY              SS  CC          HH      HH  KK    KK    PP          RR  RR      OO      OO
        SS      YY              SS  CC          HH      HH  KK    KK    PP          RR  RR      OO      OO
        SS      YY              SS  CC          HH      HH  KK      KK  PP          RR    RR    OO      OO      ....
        SS      YY              SS  CC          HH      HH  KK      KK  PP          RR    RR    OO      OO      ....
SSSSSSSS        YY      SSSSSSSS      CCCCCCCC  HH      HH  KK      KK  PP          RR      RR    OOOOOO        ....
SSSSSSSS        YY      SSSSSSSS      CCCCCCCC  HH      HH  KK      KK  PP          RR      RR    OOOOOO        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
    (2)      131        LIBRARY STRUCTURE DEFINITIONS AND MACROS
    (3)      167        LOCAL CONSTANTS AND FLAGS
    (4)      210        ITEM CODE TABLES
    (5)      254        EXE$CHKPRO - GENERAL PROTECTION ALGORITHM
    (6)      301        $CHKPRO SYSTEM SERVICE INITIAL SETUP
    (7)      388        $CHKPRO SYSTEM SERVICE ITEM SCANNING
    (8)      658        EXE$CHKPRO_INT - $CHKPRO INTERNAL ENTRY POINT
    (9)      910        EXE$GET_AUDIT - SEARCH FOR SECURITY AUDIT ACE IN THE ACL
    (10)    1017        EXE$CHECKACL - CHECK FOR AN ACE IN AN ACL
    (11)    1114        EXE$SEARCH_RIGHT - SEARCH RIGH;S DESCRIPTOR FOR AN IDENTIFIER
    (12)    1208        EXE$FINDACL - SEARCH FOR A PARTICULAR ACE IN THE ACL
    (13)    1282        EXE$CHECKPROT_16 - DO STANDARD SOGW CHECK WITH WORD INPUT
    (14)    1338        EXE$CHECKPROT - DO STANDARD SOGW PROTECTION CHECK
    (15)    1449        EXE$CHECKACMODE - DO ACCESS MODE PROTECTION CHECK
    (16)    1516        EXE$CHECKCLASS - DO NON-DISCRETIONARY SECURITY CHECK
    (17)    1614        EXE$CHECK_BYPASS - CHECK FOR BYPASS OR READALL PRIVILEGES
```

```
0000            .TITLE  SYSCHKPRO - CENTRAL PROTECTION CHECK ALGORITHM
0000            .IDENT  'V04-000'
0000            .ENABL  DBG
0000
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000
0000    ;++
0000    ; FACILITY:     VAX/VMS Exec
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       This module contains the routines that implement the protection
0000    ;       check algorithms used within VMS (UIC protection, Access Control
0000    ;       Lists, Classification mask check, etc.)
0000    ;
0000    ; ENVIRONMENT:
0000    ;
0000    ;       VAX/VMS Exec.
0000    ;
0000    ;--
0000    ;
0000    ; AUTHOR:  L. Mark Pilant,              CREATION DATE:  18-Feb-1983
0000    ;
0000    ;       (With thanks to A. Goldstein)
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-023 LMP0293         L. Mark Pilant,         2-Aug-1984  12:16
0000    ;               Clear the local ACL_PRESENT flag if SS$_IVACL is returned
0000    ;               from EXE$CHECKACL, so that EXE$GET_AUDIT is not called.
0000    ;
0000    ;       V03-022 LMP0286         L. Mark Pilant,         26-Jul-1984  12:49
0000    ;               Fix a broken intermediate branch.
0000    ;
0000    ;       V03-021 ACG0440         Andrew C. Goldstein,    23-Jul-1984  13:42
0000    ;               Add classification valid flag to ORB; use GRPPRV only with
```

```
0000  ;                UIC format owner ID
0000  ;
0000  ;       V03-020 LMP0264         L. Mark Pilant,         26-Jun-1984  13:49
0000  ;               Check for SS$_IVACL returning from EXE$CHECKACL.
0000  ;
0000  ;       V03-019 LMP0249         L. Mark Pilant,         7-May-1984  8:51
0000  ;               Modify EXE$GET_AUDIT to handle an ACL queue correctly.
0000  ;
0000  ;       V03-018 LMP0245         L. Mark Pilant,         1-May-1984  16:57
0000  ;               Remove the reference to R10 within the ACL segment scanning
0000  ;               loop.  This bug caused the segment count to be used as the
0000  ;               CHPCTL block address.
0000  ;
0000  ;       V03-017 LMP0242         L. Mark Pilant,         27-Apr-1984  14:19
0000  ;               Allow the BYPASS privilege to override SS$_IVACL.
0000  ;
0000  ;       V03-016 LMP0239         L. Mark Pilant,         23-Apr-1984  9:15
0000  ;               Add a common return point so that the block allocated from
0000  ;               the P1 lookaside list may be returned.
0000  ;
0000  ;       V03-015 TMH0015         Tim Halvorsen           14-Apr-1984
0000  ;               Fix V03-014 to define entry point as EXE$CHKPRO, not SYS$CHKPRO.
0000  ;
0000  ;       V03-014 LMP0221         L. Mark Pilant,         7-Apr-1984  14:55
0000  ;               Add support for the new internal interface.
0000  ;
0000  ;       V03-013 LMP0215         L. Mark Pilant,         21-Mar-1984  14:01
0000  ;               Make sure that the SYSTEM and OWNER protection fields have
0000  ;               control access when going from a word to the vector.
0000  ;
0000  ;       V03-012 LMP0214         L. Mark Pilant,         21-Mar-1984  11:51
0000  ;               Change EXE$CHECKPROT_16 to use the address of the protection
0000  ;               word, rather that the protection word itself.
0000  ;
0000  ;       V03-011 LMP0199         L. Mark Pilant,         28-Feb-1984  12:40
0000  ;               Correctly handle an ACL segment padded with zero.
0000  ;
0000  ;       V03-010 ACG0392         Andrew C. Goldstein,    19-Jan-1984  21:21
0000  ;               Add match-all identifier
0000  ;
0000  ;       V03-009 ACG0394         Andrew C. Goldstein,    24-Jan-1984  19:55
0000  ;               Fix loop exit in ACL check (bug in ACG0384)
0000  ;
0000  ;       V03-008 ACG0384         Andrew C. Goldstein,    19-Dec-1983  16:22
0000  ;               Allow SYSTEM and OWNER access to override ACL
0000  ;
0000  ;       V03-007 LMP0177         L. Mark Pilant,         7-Dec-1983  12:42
0000  ;               Enable the conditional non-discretionary classification
0000  ;               check by uncommenting the line.  Also change the location
0000  ;               of the flag from EXE$GL_FLAGS to EXE$GL_DYNAMIC_FLAGS.
0000  ;
0000  ;       V03-006 ACG0354         Andrew C. Goldstein,    29-Aug-1983  14:33
0000  ;               General code cleanup and tightening, add CONTROL access
0000  ;               via READALL privilege. Remove CHP$_ACCESSRIGHTS item.
0000  ;
0000  ;       V03-005 LMP0145         L. Mark Pilant,         25-Aug-1983  11:34
0000  ;               Ignore default ACEs during an ACL protection check.
```

```
0000  ;
0000  ;        V03-004 LMPBUILD        L. Mark Pilant,         28-Jun-1983  11:32
0000  ;                Fix a broken branch.
0000  ;
0000  ;        V03-003 LMP0115         L. Mark Pilant,         19-May-1983  10:38
0000  ;                Miscellaneous fixes.
0000  ;
0000  ;        V03-002 LMP0109         L. Mark Pilant,         30-Apr-1983  1:58
0000  ;                Add logic to enable the access allowed to be returned.
0000  ;                Also, several miscellaneous minor bugs were fixed.
0000  ;
0000  ;        V03-001 LMP0106         L. Mark Pilant,         26-Apr-1983  16:39
0000  ;                Change register usage in EXE$SEARCH_RIGHT.
0000  ;
0000  ;**
```

```
0000          .SBTTL  LIBRARY STRUCTURE DEFINITIONS AND MACROS
0000
0000          $ACEDEF                 ; access control list entry
0000          $ACLDEF                 ; ACL segment structure offsets
0000          $ARBDEF                 ; Agent's rights block
0000          $ARMDEF                 ; access bitmask definitions
0000          $CHPDEF                 ; service item codes
0000          $CHPCTLDEF              ; CHKPRO control block offsets
0000          $CHPRETDEF              ; $CHKPRO return arg block offsets
0000          $CLSDEF                 ; non-discretionary classification mask
0000          $DSCDEF                 ; string descriptor
0000          $IPLDEF                 ; Priority levels
0000          $ORBDEF                 ; Object's rights block
0000          $PCBDEF                 ; process control block
0000          $PRDEF                  ; Processor registers
0000          $PRBDEF                 ; internal structure protection block
0000          $PRVDEF                 ; privilege bits
0000          $PSLDEF                 ; PSL fields
0000          $SSDEF                  ; system statue codes
0000          $UICDEF                 ; UIC and identifier format
0000
0000  ; Macro to generate the necessary table entries based upon the item code.
0000
0000          .MACRO  TABLE_ENTRY     CODE, SIZE, INDEX, OFFSET
0000          TMP_PC= .
0000          .=      MIN_SIZE_TABLE+CODE
0000          .BYTE   SIZE
0000          .=      INDEX_TABLE+CODE
0000          .BYTE   INDEX
0000          .=      OFFSET_TABLE+<CODE*4>
0000          .LONG   0
0000          .=      OFFSET_TABLE+<CODE*4>+INDEX
0000          .BYTE   OFFSET
0000          .=      TMP_PC
0000          .ENDM   TABLE_ENTRY;    CODE, SIZE, INDEX, OFFSET
```

```
                  0000          .SBTTL  LOCAL CONSTANTS AND FLAGS
                  0000
        00000014  0000          MAX_ACL_DESC=   20                ; maximum number of acl segment descrs
        0000000B  0000          MAX_RIGHT_DESC= 11                ; maximum number of rights segment descrs
                  0000                                            ; (actually one less, since the list
                  0000                                            ; must be zero terminated)
        00000012  0000          MAX_CHP_CODE=   CHP$_MAX_CODE-1
                  0000
                  0000  ; Define the index values used to determine the address of the local
                  0000  ; protection structure and the offset into that structure.
                  0000
        00000000  0000          ARB_INDEX=      0
        00000001  0000          ORB_INDEX=      1
        00000002  0000          CHPCTL_INDEX=   2
        00000003  0000          CHPRET_INDEX=   3
                  0000
                  0000  ; Define the local block used when processing the user's item list.
                  0000
        00000000  0000          STRUCT_ADDR=    0                 ; Protection structure address table
        00000010  0000          LOCAL_ARB=      16                ; Agent's rights block
                  0000          ASSUME  ARB$C_HEADER EQ ARB$L_RIGHTSLIST+ARB$S_RIGHTSLIST
        00000030  0000          RIGHTS_LIST=    LOCAL_ARB+ARB$L_RIGHTSLIST      ; Agent's rights list
        0000005C  0000          LOCAL_ORB=      RIGHTS_LIST+<MAX_RIGHT_DESC*4>  ; Object's rights block
        000000B4  0000          LOCAL_CHPCTL=   LOCAL_ORB+ORB$C_LENGTH  ; Control block
        000000C0  0000          LOCAL_CHPRET=   LOCAL_CHPCTL+CHPCTL$C_LENGTH    ; Return arg block
        000000E8  0000          PRIVS_USED=     LOCAL_CHPRET+CHPRET$C_LENGTH    ; Privs used storage
        000000EC  0000          ACL_LIST=       PRIVS_USED+4    ; ACL segment descr list
        0000018C  0000          RIGHTS_DESC=    ACL_LIST+<MAX_ACL_DESC*DSC$C_S_BLN>     ; Rights list descri
                  0000
        000001E4  0000          LOCAL_LENGTH=   RIGHTS_DESC+<MAX_RIGHT_DESC*DSC$C_S_BLN>
                  0000                                            ; Length of the local storage block
                  0000          ASSUME  LOCAL_LENGTH LE 512       ; Must be less than a page
                  0000
                  0000  ; Local flags used in EXE$CHKPRO_INT.
                  0000
        00000000  0000          CHKPRO_V_ACL_PRESENT=   0               ; ACL is present
        00000001  0000          CHKPRO_V_INTERNAL=      1               ; internal vs system service entry
        00000002  0000          CHKPRO_V_NO_CHPRET=     2               ; CHPRET block not supplied
                  0000
        00000001  0000          CHKPRO_M_ACL_PRESENT=   1@CHKPRO_V_ACL_PRESENT  ; ACL is present
        00000002  0000          CHKPRO_M_INTERNAL=      1@CHKPRO_V_INTERNAL     ; internal vs system service
        00000004  0000          CHKPRO_M_NO_CHPRET=     1@CHKPRO_V_NO_CHPRET    ; CHPRET block not supplied
```

```
                 0000            .SBTTL  ITEM CODE TABLES
                 0000
             00000000            .PSECT  Y$EXEPAGED
                 0000
                 0000    ; The following table defines the minimum sizes for the various item codes.
                 0000
                 0000    MIN_SIZE_TABLE:
        00000013 0000            .BLKB   CHP$_MAX_CODE
                 0013
                 0013    ; The following table define the index associated with the local protection
                 0013    ; structure, based upon the item code.
                 0013
                 0013    INDEX_TABLE:
        00000026 0013            .BLKB   CHP$_MAX_CODE
                 0026
                 0026    ; The following table defines the offsets into the various protection
                 0026    ; structures.  The table is organized such that there are four offset bytes.
                 0026    ; These are for the ARB, ORB, CHKCTL, and CHPRET blocks in that order.
                 0026
                 0026    OFFSET_TABLE:
        00000072 0026            .BLKB   CHP$_MAX_CODE*4
                 0072
                 0072    ; Now fill the tables defined above.
                 0072
                 0072            TABLE_ENTRY     CHP$_END,       0,      0,              0
                 0072            TABLE_ENTRY     CHP$_ACCESS,    4,      CHPCTL_INDEX,   CHPCTL$L_ACCESS
                 0072            TABLE_ENTRY     CHP$_FLAGS,     4,      CHPCTL_INDEX,   CHPCTL$L_FLAGS
                 0072            TABLE_ENTRY     CHP$_PRIV,      8,      ARB_INDEX,      ARB$Q_PRIV
                 0072            TABLE_ENTRY     CHP$_ACMODE,    1,      CHPCTL_INDEX,   CHPCTL$B_MODE
                 0072            TABLE_ENTRY     CHP$_ACCLASS,   20,     ARB_INDEX,      ARB$R_CLASS
                 0072            TABLE_ENTRY     CHP$_RIGHTS,    8,      ARB_INDEX,      ARB$L_RIGHTSLIST
                 0072            TABLE_ENTRY     CHP$_ADDRIGHTS, 8,      ARB_INDEX,      ARB$L_RIGHTSLIST
                 0072            TABLE_ENTRY     CHP$_MODE,      1,      ORB_INDEX,      ORB$B_MODE
                 0072            TABLE_ENTRY     CHP$_MODES,     8,      ORB_INDEX,      ORB$Q_MODE_PROT
                 0072            TABLE_ENTRY     CHP$_MINCLASS,  20,     ORB_INDEX,      ORB$R_MIN_CLASS
                 0072            TABLE_ENTRY     CHP$_MAXCLASS,  20,     ORB_INDEX,      ORB$R_MAX_CLASS
                 0072            TABLE_ENTRY     CHP$_OWNER,     4,      ORB_INDEX,      ORB$L_OWNER
                 0072            TABLE_ENTRY     CHP$_PROT,      2,      ORB_INDEX,      ORB$W_PROT
                 0072            TABLE_ENTRY     CHP$_ACL,       4,      ORB_INDEX,      ORB$L_ACL_DESC
                 0072            TABLE_ENTRY     CHP$_AUDITNAME, 1,      CHPRET_INDEX,   CHPRET$W_AUDITLEN
                 0072            TABLE_ENTRY     CHP$_ALARMNAME, 1,      CHPRET_INDEX,   CHPRET$W_ALARMLEN
                 0072            TABLE_ENTRY     CHP$_MATCHEDACE,4,      CHPRET_INDEX,   CHPRET$W_MATCHED_ACE
                 0072            TABLE_ENTRY     CHP$_PRIVUSED,  4,      CHPRET_INDEX,   CHPRET$L_PRIVS_USED
```

```
          0072        .SBTTL  EXE$CHKPRO - GENERAL PROTECTION ALGORITHM
          0072
          0072  ;++
          0072  ;
          0072  ; FUNCTIONAL DESCRIPTION:
          0072  ;
          0072  ;       This routine implements the $CHKPRO system service, which
          0072  ;       serves as a centralized protection check. Depending on the
          0072  ;       items supplied, the following forms of protection check
          0072  ;       are available:
          0072  ;               access mode
          0072  ;               non-discretionary classification
          0072  ;               access control list
          0072  ;               simple SOGW mask
          0072  ;               audit log and alarm
          0072  ;
          0072  ; CALLING SEQUENCE:
          0072  ;       EXE$CHKPRO (ITEM_LIST)
          0072  ;
          0072  ; INPUT PARAMETERS:
          0072  ;       ITEM_LIST: address of item descriptor list
          0072  ;
          0072  ; IMPLICIT INPUTS:
          0072  ;       SCH$GL_CURPCB: PCB address of process
          0072  ;       previous access mode (access mode of caller)
          0072  ;
          0072  ; OUTPUT PARAMETERS:
          0072  ;       ITEM_LIST: address of item descriptor list
          0072  ;
          0072  ; IMPLICIT OUTPUTS:
          0072  ;       NONE
          0072  ;
          0072  ; ROUTINE VALUE:
          0072  ;       SS$_NORMAL: access granted
          0072  ;       SS$_NOPRIV: access denied
          0072  ;       SS$_ACCVIO: item list or item buffers inaccessible
          0072  ;
          0072  ; SIDE EFFECTS:
          0072  ;       NONE
          0072  ;
          0072  ;--
          0072
          0072  ; Define the offsets into the routine argument list
          0072
00000000  0072          CHKPRO_ARGCOUNT=        0
00000004  0072          CHKPRO_ITMLST=          4
```

```
                              0072         .SBTTL  $CHKPRO SYSTEM SERVICE INITIAL SETUP
                              0072
                              0072  ; Within the main body of the protection checking routine (i.e., the item
                              0072  ; descriptor scanner), the following register conventions are used:
                              0072  ;
                              0072  ;        R11 -   address of the local storage block
                              0072  ;        R10 -   address of the current item list descriptor
                              0072  ;        R9 -    return length storage address
                              0072  ;        R8 -    input/output buffer address
                              0072  ;        R7 -    size of the input/output buffer
                              0072  ;        R6 -    index into rights descriptor list
                              0072  ;        R5 -    address for item in local protection structure
                              0072
                              0072         .ENABLE LSB
                              0072
                        0FFC  0072         .ENTRY  EXE$CHKPRO,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                              0074
                              0074  ; Local storage block from P1 lookaside list.
                              0074
      5B   00000000'FF    0F  0074         REMQUE  @CTL$GL_KRPFL,R11        ; Else allocate from P1 lookaside list
                   04     1C  007B         BVC     5$                       ; Xfer if able to get one
                              007D         BUG_CHECK       KRPEMPTY         ; Else come to a screeching halt
                              0081  ;
                              0081  ; Set up the initial defaults in the local protection structures.
                              0081  ;
 6B   01E4 8F   00   6E   00  2C  0081  5$:    MOVC5   #0,(SP),#0,#LOCAL_LENGTH,(R11)  ; Initially clear out the block
                              0089
                              0089  ; Set up ARB defaults.
                              0089
      50   00000000'9F    D0  0089         MOVL    @#SCH$GL_CURPCB,R0       ; get current PCB address
   10 AB   008C D0   30   28  0090         MOVC3   #ARB$C_HEADER,@PCB$L_ARB(R0),LOCAL_ARB(R11)      ; Copy minimal ARB
        50    10 AB       9E  0097         MOVAB   LOCAL_ARB(R11),R0        ; Set address of protection structure
             6B    50     D0  009B         MOVL    R0,STRUCT_ADDR(R11)      ; Save ARB address for later
                   56     D4  009E         CLRL    R6                       ; Reset rights list segment index
              20 A046     D5  00A0  10$:   TSTL    ARB$L_RIGHTSLIST(R0)[R6]         ; End of the list?
                   04     13  00A4         BEQL    15$                      ; Xfer if so, index now set
                   56     D6  00A6         INCL    R6                       ; Else up the index
                   F6     11  00A8         BRB     10$                      ; And try the next one
                              00AA
                              00AA  ; Set up ORB defaults.
                              00AA
        50    5C AB       9E  00AA  15$:   MOVAB   LOCAL_ORB(R11),R0        ; Set address of protection structure
        04 AB    50       D0  00AE         MOVL    R0,STRUCT_ADDR+4(R11)    ; Save ORB address for later
        10 A0    04       D0  00B2         MOVL    #4,ORB$B_MODE(R0)        ; Default access mode of object
                              00B6
                              00B6  ; Set up CHPCTL block defaults.
                              00B6
      50    0084 CB       9E  00B6         MOVAB   LOCAL_CHPCTL(R11),R0     ; Set address of protection structure
        08 AB    50       D0  00BB         MOVL    R0,STRUCT_ADDR+8(R11)    ; Save CHPCTL address for later
        04 A0    03       D0  00BF         MOVL    #CHPCTL$M_READ!CHPCTL$M_WRITE,CHPCTL$L_FLAGS(R0)
                              00C3                                          ; allowing for read and write access
                   51     DC  00C3         MOVPSL  R1                       ; Get the current PSL
   08 A0    51   02   16  EF  00C5         EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,CHPCTL$B_MODE(R0)
                              00CB                                          ; get accessor mode
                              00CB
                              00CB  ; Set up the CHPRET block defaults.
                              00CB
      50   00C0 CB   9E  00CB   358         MOVAB   LOCAL_CHPRET(R11),R0     ; Set address of protection structure
        0C AB    50  D0  00D0   359         MOVL    R0,STRUCT_ADDR+12(R11)   ; Save CHPRET address for later
   24 A0   00E8 CB   9E  00D4   360         MOVAB   PRIVS_USED(R11),CHPRET$L_PRIVS_USED(R0) ; Where to return privs used
                         00DA   361
                         00DA   362 ; Start the item list processing.
                         00DA   363
        5A   04 AC   D0  00DA   364         MOVL    CHKPRO_ITMLST(AP),R10    ; set the address of the item list
                         00DE   365         IFRD    #4,(R10),GET_ITEM        ; probe first longword of item list
                         00E4   366
                         00E4   367 ; Error returns.
                         00E4   368
                         00E4   369 RETURN_ACCVIO:
           50   0C   D0  00E4   370         MOVL    #SS$_ACCVIO,R0           ; set error status
              0352   31  00E7   371         BRW     RETURN_P1_BLOCK          ;   and return
                         00EA   372
                         00EA   373 BADPARAM:
           50   14   D0  00EA   374         MOVL    #SS$_BADPARAM,R0         ; set status
              034C   31  00ED   375         BRW     RETURN_P1_BLOCK          ; and return
                         00F0   376 ;
                         00F0   377 ; To here when all of the item descriptors have been processed.  Now begin
                         00F0   378 ; the actual protection checking.  This consists of calling a series of
                         00F0   379 ; routines to do the various checks.
                         00F0   380 ;
                         00F0   381 FINISH_ITEMS:
                5B   DD  00F0   382         PUSHL   R11                      ; Save address of the local storage block
                57   D4  00F2   383         CLRL    R7                       ; Reset all flags & indicate service entry
           50   6B   7D  00F4   384         MOVQ    STRUCT_ADDR(R11),R0      ; Get ARB and ORB addresses
        52   08 AB   7D  00F7   385         MOVQ    STRUCT_ADDR+8(R11),R2    ; Now for CHPCTL and CHPRET addresses
              0184   31  00FB   386         BRW     EXE$CHKPRO_CMN           ; join common code
```

```
                      00FE          .SBTTL  $CHKPRO SYSTEM SERVICE ITEM SCANNING
                      00FE   ;
                      00FE   ; Scan through the item list, acquiring the input information as encountered.
                      00FE   ;
                      00FE   GET_ITEM:
        57    8A   3C 00FE           MOVZWL  (R10)+,R7                 ; get next item length
              ED   13 0101           BEQL    FINISH_ITEMS              ; if zero, end of list
        54    8A   3C 0103           MOVZWL  (R10)+,R4                 ; get item code
        12    54   D1 0106           CMPL    R4,#MAX_CHP_CODE          ; range check item code
              DF   1A 0109           BGTRU   BADPARAM
51    FEF0 CF44    9A 010B           MOVZBL  MIN_SIZE_TABLE[R4],R1     ; get minimum size allowed
        51    57   D1 0111           CMPL    R7,R1                     ; less than the min required?
              D4   1F 0114           BLSSU   BADPARAM                  ; xfer if so
                      0116           IFNORD  #12,(R10),RETURN_ACCVIO1        ; probe rest of item + start of next
        58    8A   D0 011C           MOVL    (R10)+,R8                 ; get buffer address
        51    57   D0 011F           MOVL    R7,R1                     ; copy buffer descriptor
        50    58   D0 0122           MOVL    R8,R0
              53   D4 0125           CLRL    R3                        ; use prev mode only
   00000000'EF     16 0127           JSB     EXE$PROBER                ; and probe buffer for readability
           52 50   E9 012D           BLBC    R0,RETURN_ACCVIO1         ; branch on failure
        59    8A   D0 0130           MOVL    (R10)+,R9                 ; get the address to return length
              08   13 0133           BEQL    30$                       ; xfer if no return length required
                      0135           IFNOWRT #2,(R9),RETURN_ACCVIO1    ; else check for write access
              69   B4 013B           CLRW    (R9)                      ; preset to zero
                      013D
                      013D   ; Use the index obtained from the index table to get the local protection
                      013D   ; structure base address and the offset into that same structure.
                      013D
50    FED1 CF44    9A 013D   30$:    MOVZBL  INDEX_TABLE[R4],R0        ; Get appropriate index table entry
     53   6B40     D0 0143           MOVL    STRUCT_ADDR(R11)[R0],R3 ; Get structure base address
51    FEDA CF44    DE 0147           MOVAL   OFFSET_TABLE[R4],R1       ; Get offset table entry
     55   6140     9A 014D           MOVZBL  (R1)[R0],R5               ; Get the offset
        55    53   C0 0151           ADDL    R3,R5                     ; compute protection structure field address
                      0154   ;
                      0154   ; All of the basic checks about the item descriptor have succeeded.  Now
                      0154   ; dispatch based upon the item code to take the appropriate action.
                      0154   ;
   11    01    54  CF 0154           CASEL   R4,#1,#MAX_CHP_CODE-1
                0065' 0158   40$:    .WORD   ITEM_ACCESS-40$           ; CHP$_ACCESS
                0065' 015A           .WORD   ITEM_FLAGS-40$            ; CHP$_FLAGS
                0057' 015C           .WORD   ITEM_PRIV-40$             ; CHP$_PRIV
                006A' 015E           .WORD   ITEM_ACMODE-40$           ; CHP$_ACMODE
                003F' 0160           .WORD   ITEM_ACCLASS-40$          ; CHP$_ACCLASS
                00C4' 0162           .WORD   ITEM_RIGHTS-40$           ; CHP$_RIGHTS
                00C4' 0164           .WORD   ITEM_ADDRIGHTS-40$        ; CHP$_ADDRIGHTS
                009A' 0166           .WORD   ITEM_MODE-40$             ; CHP$_MODE
                0057' 0168           .WORD   ITEM_MODES-40$            ; CHP$_MODES
                0030' 016A           .WORD   ITEM_MINCLASS-40$         ; CHP$_MINCLASS
                0036' 016C           .WORD   ITEM_MAXCLASS-40$         ; CHP$_MAXCLASS
                0065' 016E           .WORD   ITEM_OWNER-40$            ; CHP$_OWNER
                00EF' 0170           .WORD   ITEM_PROT-40$             ; CHP$_PROT
                00A1' 0172           .WORD   ITEM_ACL-40$              ; CHP$_ACL
                0081' 0174           .WORD   ITEM_AUDITNAME-40$        ; CHP$_AUDITNAME
                0081' 0176           .WORD   ITEM_ALARMNAME-40$        ; CHP$_ALARMNAME
                0081' 0178           .WORD   ITEM_MATCHEDACE-40$       ; CHP$_MATCHEDACE
                006F' 017A           .WORD   ITEM_PRIVUSED-40$         ; CHP$_PRIVUSED
                      017C   ;
```

```
                    017C  ; Falling through indicates a bad parameter.
                    017C  ;
                    017C  BADPARAM1:
      50    14  D0  017C          MOVL    #SS$_BADPARAM,R0        ; set status
        02BA    31  017F          BRW     RETURN_P1_BLOCK         ; and return
                    0182  ;
                    0182  ; What to do when some portion of the item descriptor cannot be read or
                    0182  ; written as necessary.
                    0182  ;
                    0182  RETURN_ACCVIO1:
      50    0C  D0  0182          MOVL    #SS$_ACCVIO,R0          ; set error status
        02B4    31  0185          BRW     RETURN_P1_BLOCK         ;   and return
                    0188  ;
                    0188  ; Common routines to copy item text into the local storage block.  For all
                    0188  ; of the ITEM_xxx routines below, the following register usage is utilized:
                    0188  ;
                    0188  ;       R0-R2   Scratch
                    0188  ;       R3      Address of the local protection structure
                    0188  ;       R4      Item code
                    0188  ;       R5      Address of the local protection structure field
                    0188
                    0188  ;
                    0188  ; Classification mask item. For the first of MIN or MAX class, copy
                    0188  ; the item into its partner to default the contents.
                    0188  ;
                    0188  ITEM_MINCLASS:
    50   44 A3  9E  0188          MOVAB   ORB$R_MAX_CLASS(R3),R0  ; Point to other mask
            04  11  018C          BRB     43$
                    018E  ITEM_MAXCLASS:
    50   30 A3  9E  018E          MOVAB   ORB$R_MIN_CLASS(R3),R0  ; Point to other mask
03 0B A3    04  E3  0192  43$:    BBCS    #ORB$V_CLASS_PROT,ORB$B_FLAGS(R3),44$ ; Mark classification present
                    0197  ITEM_ACCLASS:
                    0197          ASSUME  ARB$S_CLASS EQ 20
                    0197          ASSUME  ORB$S_MIN_CLASS EQ 20
                    0197          ASSUME  ORB$S_MAX_CLASS EQ 20
      50    55  D0  0197          MOVL    R5,R0                   ; Copy mask address
      80    68  7D  019A  44$:    MOVQ    (R8),(R0)+              ; First 8 bytes
      85    88  7D  019D          MOVQ    (R8)+,(R5)+             ; First 8 bytes
      80    68  7D  01A0          MOVQ    (R8),(R0)+              ; Second 8 bytes
      85    88  7D  01A3          MOVQ    (R8)+,(R5)+             ; Second 8 bytes
      80    68  D0  01A6          MOVL    (R8),(R0)+              ; Final 4 bytes
      85    88  D0  01A9          MOVL    (R8)+,(R5)+             ; Final 4 bytes
                    01AC  NEXT_ITEM:
        FF4F    31  01AC          BRW     GET_ITEM                ; Go get next item
                    01AF  ;
                    01AF  ; Quadword item.
                    01AF  ;
                    01AF  ITEM_PRIV:
                    01AF  ITEM_MODES:
      65    68  7D  01AF          MOVQ    (R8),(R5)               ; store in local protection structure
      09    54  D1  01B2          CMPL    R4,#CHP$_MODES          ; Mode protection vector?
            04  12  01B5          BNEQ    45$                     ; Xfer if not
   0B A3    04  88  01B7          BISB2   #ORB$M_MODE_VECTOR,ORB$B_FLAGS(R3)      ; Else note use of vector
            EF  11  01BB  45$:    BRB     NEXT_ITEM               ; Go get the next item in the list
                    01BD  ;
                    01BD  ; Longword item.
                    01BD  ;
```

```
                         01BD   ITEM_ACCESS:
                         01BD   ITEM_FLAGS:
                         01BD   ITEM_OWNER:
          65   68   D0   01BD           MOVL    (R8),(R5)               ; store in local protection structure
               EA   11   01C0           BRB     NEXT_ITEM               ; go get next item_descriptor
                         01C2   ;
                         01C2   ; Byte item.
                         01C2   ;
                         01C2   ITEM_ACMODE:
          65   68   9A   01C2           MOVZBL  (R8),(R5)               ; store in local protection structure
               E5   11   01C5           BRB     NEXT_ITEM               ; go get next item_descriptor
                         01C7   ;
                         01C7   ; Common path for returning a longword value of some sort.  Check for write
                         01C7   ; accessibility, and then save the return address.
                         01C7   ;
                         01C7   ITEM_PRIVUSED:
                         01C7           IFNOWRT #4,(R8),RETURN_ACCVIO1  ; xfer if cannot be written
          65   58   D0   01CD           MOVL    R8,(R5)                 ; where to return information
               59   D5   01D0           TSTL    R9                      ; return length needed?
               D8   13   01D2           BEQL    NEXT_ITEM               ; xfer if not
          69   04   B0   01D4           MOVW    #4,(R9)                 ; else set return length
               D3   11   01D7           BRB     NEXT_ITEM               ; go get next item descriptor
                         01D9   ;
                         01D9   ; Common path for returning a descriptor of some sort.  Check for write
                         01D9   ; accessibility, and then save the needed arguments.
                         01D9   ;
                         01D9           ASSUME  CHPRET$W_AUDITLEN EQ CHPRET$L_AUDIT-4
                         01D9           ASSUME  CHPRET$L_AUDITRET EQ CHPRET$L_AUDIT+4
                         01D9           ASSUME  CHPRET$W_ALARMLEN EQ CHPRET$L_ALARM-4
                         01D9           ASSUME  CHPRET$L_ALARMRET EQ CHPRET$L_ALARM+4
                         01D9           ASSUME  CHPRET$W_MATCHED_ACELEN EQ CHPRET$L_MATCHED_ACE-4
                         01D9           ASSUME  CHPRET$L_MATCHED_ACERET EQ CHPRET$L_MATCHED_ACE+4
                         01D9
                         01D9   ITEM_AUDITNAME:
                         01D9   ITEM_ALARMNAME:
                         01D9   ITEM_MATCHEDACE:
          51   57   D0   01D9           MOVL    R7,R1                   ; copy buffer descriptor
          50   58   D0   01DC           MOVL    R8,R0
               53   D4   01DF           CLRL    R3                      ; use prev mode only
    00000000'EF     16   01E1           JSB     EXE$PROBEW              ; check item descr for writing
           98  50   E9   01E7           BLBC    R0,RETURN_ACCVIO1       ; xfer if cannot be written
          85   57   7D   01EA           MOVQ    R7,(R5)+                ; save descriptor
          85   59   D0   01ED           MOVL    R9,(R5)+                ; save return address specified
               BA   11   01F0           BRB     NEXT_ITEM               ; go get the next descriptor
                         01F2   ;
                         01F2   ; Special case item handling code follows.
                         01F2   ;
                         01F2   ; Extract simple access mode.
                         01F2   ;
                         01F2   ITEM_MODE:
65   68   02   00   EF   01F2           EXTZV   #0,#2,(R8),(R5)         ; get access mode protection
               B3   11   01F7           BRB     NEXT_ITEM               ; go get next item descriptor
                         01F9   ;
                         01F9   ; Process ACL segment descriptor.
                         01F9   ;
                         01F9   ITEM_ACL:
       50   28 A3   D0   01F9           MOVL    ORB$L_ACL_COUNT(R3),R0 ; get current number of descrs
```

```
             06   12 01FD            BNEQ    50$                     ; xfer if not the first one
  2C A3    00EC CB   9E 01FF            MOVAB   ACL_LIST(R11),ORB$L_ACL_DESC(R3)        ; Else note address
        14   50   D1 0205   50$:     CMPL    R0,#MAX_ACL_DESC        ; table full?
             0A   1E 0208            BGEQU   60$                     ; xfer if so
   2C B340   57   7D 020A            MOVQ    R7,@ORB$L_ACL_DESC(R3)[R0]      ; else save another
        28 A3   D6 020F            INCL    ORB$L_ACL_COUNT(R3)     ; up count of ACL segments
             98   11 0212            BRB     NEXT_ITEM               ; go get next item descriptor
                     0214
     50    09F8 8F   3C 0214   60$:     MOVZWL  #SS$_ACLFULL,R0         ; set error code
           0220   31 0219            BRW     RETURN_P1_BLOCK         ; and return
                     021C   ;
                     021C   ; Build specified rights list.
                     021C   ;
                     021C   ITEM_RIGHTS:
                     021C   ITEM_ADDRIGHTS:
                     021C   ;
                     021C   ; If a new rights list is specified, forget any existing entries.
                     021C   ;
        06   54   D1 021C   90$:     CMPL    R4,#CHP$_RIGHTS         ; see if new rights list specified
             02   12 021F            BNEQ    100$                    ; branch if not - add to existing
             56   D4 0221            CLRL    R6                      ; initialize counter
                     0223   ;
                     0223   ; Add a new rights list descriptor to any that already exist.
                     0223   ;
        0B   56   D1 0223   100$:    CMPL    R6,#MAX_RIGHT_DESC      ; is there room for this descriptor?
             17   1E 0226            BGEQU   110$                    ; xfer if not, note error
  50   018C CB46   7E 0228            MOVAQ   RIGHTS_DESC(R11)[R6],R0 ; set address of descriptor
   30 AB46   50   D0 022E            MOVL    R0,RIGHTS_LIST(R11)[R6] ; save address for later
        60   57   7D 0233            MOVQ    R7,(R0)                 ; save descriptor for later
             56   D6 0236            INCL    R6                      ; next available
      30 AB46   D4 0238            CLRL    RIGHTS_LIST(R11)[R6]    ; mark current end
                     023C   NEXT_ITEM1:
           FEBF   31 023C            BRW     GET_ITEM                ; go get next item descriptor
                     023F
     50    09E8 8F   3C 023F   110$:    MOVZWL  #SS$_RIGHTSFULL,R0      ; else set error code
           01F5   31 0244            BRW     RETURN_P1_BLOCK         ; and exit stage left
                     0247
                     0247   ;
                     0247   ; The following section of code converts the standard protection mask into
                     0247   ; a series of longwords, each representing a specific class of users (system,
                     0247   ; group, etc.)  It further assumes that any extensions to the protection mask
                     0247   ; will be in 4 bit chunks;  thus adding an additional word.  Following is
                     0247   ; a diagram of the mapping that takes place:
                     0247   ;
                     0247   ;               +---------+---------+---------+---------+
                     0247   ;               !  WORLD  !  GROUP  !  OWNER  !  SYSTEM !
                     0247   ;               +---------+---------+---------+---------+
                     0247   ;               ! D!E!W!R ! D!E!W!R ! D!E!W!R ! D!E!W!R !
                     0247   ;               +---------+---------+---------+---------+
                     0247   ;               15                                      0
                     0247   ;
                     0247   ;               +---------+---------+---------+---------+
                     0247   ;               !  WORLD  !  GROUP  !  OWNER  !  SYSTEM !
                     0247   ;               +---------+---------+---------+---------+
                     0247   ;               !   ...!C !   ...!C !   ...!C !   ...!C !
                     0247   ;               +---------+---------+---------+---------+
                     0247   ;               15                                      0
```

```
                       0247  ;                                      :
                       0247  ;                                      :
                       0247  ;                             \ :        /
                       0247  ;                              \:       /
                       0247  ;                               \      /
                       0247  ;                                \    /
                       0247  ;                                 \  /
                       0247  ;                                  \/
                       0247  ;
                       0247  ;              31                                       0
                       0247  ;               +---------------------------------------+
                       0247  ;               |                          ...|C|D|E|W|R|       SYSTEM
                       0247  ;               +---------------------------------------+
                       0247  ;               |                          ...|C|D|E|W|R|       OWNER
                       0247  ;               +---------------------------------------+
                       0247  ;               |                          ...|C|D|E|W|R|       GROUP
                       0247  ;               +---------------------------------------+
                       0247  ;               |                          ...|C|D|E|W|R|       WORLD
                       0247  ;               +---------------------------------------+
                       0247  ;              31                                       0
                       0247
                       0247
                       0247  ITEM_PROT:
 50   57   FF 8F   78  0247          ASHL    #-1,R7,R0               ; get mask size in words
         08   50   D1  024C          CMPL    R0,#8                   ; too much supplied?
              25   1B  024F          BLEQU   140$                    ; xfer if not - start loop
            FF28   31  0251          BRW     BADPARAM1               ; return error status
                       0254
              51   D4  0254  120$:   CLRL    R1                      ; else reset index
       52   6840   3C  0256          MOVZWL  (R8)[R0],R2             ; get next protection word
    53   50   02   78  025A          ASHL    #2,R0,R3                ; calc position in output mask
              54   D4  025E  130$:   CLRL    R4                      ; preload R4
 54   52   F0 8F   8B  0260          BICB3   #^XF0,R2,R4             ; get protection bits
    54   54   53   9C  0265          ROTL    R3,R4,R4                ; shift into position
      6541    54   C8  0269          BISL2   R4,(R5)[R1]             ; save in output mask
 52   52   FC 8F   9C  026D          ROTL    #-4,R2,R2               ; get next set of bits
      E8 51   04   F2  0272          AOBLSS  #4,R1,130$              ; continue till done
           DB 50   F4  0276  140$:   SOBGEQ  R0,120$                 ; go get next protection word
              C1   11  0279          BRB     NEXT_ITEM1              ; done
                       027B
                       027B          .DISABLE LSB
```

```
                               027B   658         .SBTTL  EXE$CHKPRO_INT - $CHKPRO INTERNAL ENTRY POINT
                               027B   659
                               027B   660 ;++
                               027B   661 ;
                               027B   662 ; FUNCTIONAL DESCRIPTION:
                               027B   663 ;
                               027B   664 ;       This is the internal entry point to the $CHKPRO system service.  This
                               027B   665 ;       entry point may be used to avoid the overhead associated with insuring
                               027B   666 ;       that the item list is valid.  This is done by assuming that the caller
                               027B   667 ;       has filled in the necessary arg blocks in the same manner as the item
                               027B   668 ;       list processing code above.
                               027B   669 ;
                               027B   670 ; CALLING SEQUENCE:
                               027B   671 ;       JSB     EXE$CHKPRO_INT
                               027B   672 ;
                               027B   673 ; INPUT PARAMETERS:
                               027B   674 ;       ARB     (R0): address of the agents rights block
                               027B   675 ;       ORB     (R1): address of the objects rights block
                               027B   676 ;       CHPCTL  (R2): address of the protection check control block
                               027B   677 ;       CHPRET  (R3): address of the return argument block
                               027B   678 ;
                               027B   679 ; IMPLICIT INPUTS:
                               027B   680 ;       NONE
                               027B   681 ;
                               027B   682 ; OUTPUT PARAMETERS:
                               027B   683 ;       SAME AS EXE$CHKPRO
                               027B   684 ;
                               027B   685 ; IMPLICIT OUTPUTS:
                               027B   686 ;       NONE
                               027B   687 ;
                               027B   688 ; ROUTINE VALUE:
                               027B   689 ;       SAME AS EXE$CHKPRO
                               027B   690 ;
                               027B   691 ; SIDE EFFECTS:
                               027B   692 ;       NONE
                               027B   693 ;
                               027B   694 ;--
                               027B   695
                               027B   696 ; Internal entry point to the protection check system service.
                               027B   697
                               027B   698 EXE$CHKPRO_INT::
            0FFE 8F    BB      027B   699         PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; save work regs
            57   02    D0      027F   700         MOVL    #CHKPRO_M_INTERNAL,R7           ; Reset flags & indicate internal en
                               0282   701 EXE$CHKPRO_CMN:
            58   50    7D      0282   702         MOVQ    R0,R8                           ; put structure addresses
            5A   52    7D      0285   703         MOVQ    R2,R10                          ;   in a more useful place
                 5B    D5      0288   704         TSTL    R11                             ; was a return arg block given?
                 14    12      028A   705         BNEQ    5$                              ; xfer if so, skip following
            57   04    C8      028C   706         BISL2   #CHKPRO_M_NO_CHPRET,R7          ; note fabricated CHPRET block
                               028F   707         ASSUME  <CHPRET$C_LENGTH & 3> EQ 0
            5E   2C    C2      028F   708         SUBL2   #CHPRET$C_LENGTH+4,SP           ; else make room for one
            5B   5E    D0      0292   709         MOVL    SP,R11                          ; save address
6B   2C   00   6E   00    2C      0295   710         MOVC5   #0,(SP),#0,#CHPRET$C_LENGTH+4,(R11)
       24 AB   5B   28    C1      029B   711         ADDL3   #CHPRET$C_LENGTH,R11,CHPRET$L_PRIVS_USED(R11)   ; privs used return
                               02A0   712
                               02A0   713 ; If an ACL is supplied as a queue, lock it now.
                               02A0   714
```

```
      17 0B A9    01    E1  02A0  5$:     BBC     #ORB$V_ACL_QUEUE,ORB$B_FLAGS(R9),10$      ; Skip if not a queue
                            02A5          DSBINT  #IPL$_ASTDEL                            ; Raise IPL to prevent deletion
 54    00000000'9F      D0  02AB          MOVL    @#SCH$GL_CURPCB,R4                      ; Get current PCB address
       50    04 A9      9E  02B2          MOVAB   ORB$L_ACL_MUTEX(R9),R0                  ; Set mutex address
       00000000'9F      16  02B6          JSB     @#SCH$LOCKR                             ; Lock mutex for reading
                            02BC
                            02BC  ; Set up an alternate SOGW protection vector/mask.  This is used when checking
                            02BC  ; to see if system or owner are allowed access if the ACL actually denies access.
                            02BC
                            02BC          ASSUME  ORB$L_SYS_PROT EQ ORB$W_PROT
                            02BC          ASSUME  ORB$L_OWN_PROT EQ ORB$L_SYS_PROT+4
                            02BC          ASSUME  ORB$L_WOR_PROT EQ ORB$L_GRP_PROT+4
                            02BC
            7E    01    CE  02BC  10$:    MNEGL   #1,-(SP)                                ; deny access to group
            7E    01    CE  02BF          MNEGL   #1,-(SP)                                ;   and world
         7E    18 A9    7D  02C2          MOVQ    ORB$L_SYS_PROT(R9),-(SP)                ; Original system & owner protection
      04 0B A9    00    E1  02C6          BBC     #ORB$V_PROT_16,ORB$B_FLAGS(R9),15$         ; Xfer if full vector
         01 AE    01    8E  02CB          MNEGB   #1,1(SP)                                ; Else deny group & world access
                            02CF
                            02CF  ; Perform the access mode protection check.
                            02CF
            53    6A    D0  02CF  15$:    MOVL    CHPCTL$L_ACCESS(R10),R3                 ; set up input parameters
         54    08 AA    9A  02D2          MOVZBL  CHPCTL$B_MODE(R10),R4
         55    10 A9    9A  02D6          MOVZBL  ORB$B_MODE(R9),R5                       ; assume simple mode protection
      04 0B A9    02    E1  02DA          BBC     #ORB$V_MODE_VECTOR,ORB$B_FLAGS(R9),20$  ; xfer if correct
         55    10 A9    9E  02DF          MOVAB   ORB$Q_MODE_PROT(R9),R5                  ; else set address of vector
              03E9      30  02E3  20$:    BSBW    EXE$CHECKACMODE                         ; do the actual check
              5C 50     E9  02E6          BLBC    R0,45$                                  ; xfer if access was denied
                            02E9
                            02E9  ; Next comes the non-discretionary protection check, if enabled (via a
                            02E9  ; SYSGEN flag), and if it is called for.
                            02E9
22 00000000'9F    00'   E1  02E9          BBC     S^#EXE$V_CLASS_PROT,@#EXE$GL_DYNAMIC_FLAGS,30$  ; xfer if not enable
      1D 0B A9    04    E1  02F1          BBC     #ORB$V_CLASS_PROT,ORB$B_FLAGS(R9),30$ ; xfer if not present
            52    68    9E  02F6          MOVAB   ARB$Q_PRIV(R8),R2                       ; else set up input parameters
         53    04 AA    D0  02F9          MOVL    CHPCTL$L_FLAGS(R10),R3
         54    0C A8    9E  02FD          MOVAB   ARB$R_CLASS(R8),R4
         55    30 A9    9E  0301          MOVAB   ORB$R_MIN_CLASS(R9),R5
         56    44 A9    9E  0305          MOVAB   ORB$R_MAX_CLASS(R9),R6
              03FC      30  0309          BSBW    EXE$CHECKCLASS                          ; do the check
         24 BB    51    C8  030C          BISL2   R1,@CHPRET$L_PRIVS_USED(R11)            ; note any privileges used
            32    50    E9  0310  25$:    BLBC    R0,45$                                  ; xfer if access denied
                            0313
                            0313  ; If there is any ACL, check it now.  This may be in one of two forms:
                            0313  ; 1) an ACL queue segment listhead or 2) a ACL segment descriptor vector
                            0313  ; and an associated count (of the number of descriptors).
                            0313
            7E    5A    7D  0313  30$:    MOVQ    R10,-(SP)                               ; save CHPCTL and CHPRET
            53    6A    D0  0316          MOVL    CHPCTL$L_ACCESS(R10),R3                 ; set up CHECKACL input parameters
         54    20 A8    9E  0319          MOVAB   ARB$L_RIGHTSLIST(R8),R4
      26 0B A9    01    E1  031D          BBC     #ORB$V_ACL_QUEUE,ORB$B_FLAGS(R9),50$     ; xfer if not a queue
                            0322
                            0322  ; Handle the ACL segment queue here.
                            0322
         5A    28 A9    9E  0322          MOVAB   ORB$L_ACLFL(R9),R10                     ; set address of queue head
                  6A    D5  0326          TSTL    (R10)                                   ; Is queue head valid?
                  40    13  0328          BEQL    70$                                     ; Xfer if not, nothing to check
```

```
                 5B    5A   D0  032A   772           MOVL    R10,R11                            ; Else copy address for later
                 5A    6A   D0  032D   773 40$:      MOVL    (R10),R10                          ; get address of next segment
                 5B    5A   D1  0330   774           CMPL    R10,R11                            ; end of the line?
                       35   13  0333   775           BEQL    70$                                ; xfer if so
                       55   D4  0335   776           CLRL    R5                                 ; else preset segment size
          55    08 AA   0C  A3  0337   777           SUBW3   #ACL$C_LENGTH,ACL$W_SIZE(R10),R5        ; set segment size
                56    0C AA   9E  033C   778           MOVAB   ACL$L_LIST(R10),R6                 ; set address of ACEs
                       13   11  0340   779           BRB     65$                                ; go do the ACL check
                                0342   780
                                0342   781 ; If an invalid ACL has been seen, clear the local ACL_PRESENT flag so that
                                0342   782 ; it is not checked for an AUDIT or ALARM ACE.
                                0342   783
                 57    01   CA  0342   784 44$:      BICL2   #CHKPRO_M_ACL_PRESENT,R7           ; Forget any ACL present
                                0345   785
                                0345   786 ; Intermediate branch for BYPASS checking.
                                0345   787
                     0088   31  0345   788 45$:      BRW     BYPASS_CHECK                       ; Go check for BYPASS priv
                                0348   789
                                0348   790 ; Handle the descriptor vector here.
                                0348   791
              5A    28 A9   D0  0348   792 50$:      MOVL    ORB$L_ACL_COUNT(R9),R10            ; get the number of descriptors
                       1C   13  034C   793           BEQL    70$                                ; xfer if no ACL supplied
              5B    2C A9   D0  034E   794           MOVL    ORB$L_ACL_DESC(RC),R11             ; get address of descriptor list
                 55    8B   7D  0352   795 60$:      MOVQ    (R11)‡,R5                          ; get a descriptor
                                0355   796
                                0355   797 ; Now check the ACL segment described by R5 & R6.
                                0355   798
                 57    01   C8  0355   799 65$:      BISL2   #CHKPRO_M_ACL_PRESENT,R7           ; note an ACL present
                     017D   30  0358   800           BSBW    EXE$CHECKACL                       ; search this segment
          09D8 8F    50   B1  035B   801           CMPW    R0,#SS$_NOENTRY                    ; was anything found?
                       0D   12  0360   802           BNEQ    80$                                ; xfer if so...go deal with it
          C6 0B A9   01   E0  0362   803           BBS     #ORB$V_ACL_QUEUE,ORB$B_FLAGS(R9),40$     ; if a queue, go get next
                 E8    5A   F5  0367   804           SOBGTR  R10,60$                            ; else continue with next segment
                 5A    8E   7D  036A   805 70$:      MOVQ    (SP)+,R10                          ; restore saved registers
                       33   11  036D   806           BRB     110$                               ; go try next check
                                036F   807
                                036F   808 ; If the ACL segment is invalid, go check for BYPASS.
                                036F   809
          21E4 8F    50   B1  036F   810 80$:      CMPW    R0,#SS$_IVACL                      ; Valid ACL?
                       CC   13  0374   811           BEQL    44$                                ; Xfer if not
                                0376   812
                                0376   813 ; An entry was found in the ACL.  It may grant or deny access.
                                0376   814
                 5A    8E   7D  0376   815           MOVQ    (SP)+,R10                          ; restore saved registers
                 56    50   D0  0379   816           MOVL    R0,R6                              ; Save current status
                                037C   817           ASSUME  CHPRET$W_MATCHED_ACELEN EQ CHPRET$L_MATCHED_ACE-4
              52    18 AB   7D  037C   818           MOVQ    CHPRET$W_MATCHED_ACELEN(R11),R2 ; get the return descriptor
                       12   13  0380   819           BEQL    100$                               ; xfer if no need to return
                 54    61   9A  0382   820           MOVZBL  ACE$B_SIZE(R1),R4                  ; else get size of the ACE
              55    20 AB   D0  0385   821           MOVL    CHPRET$L_MATCHED_ACERET(R11),R5 ; note where length is returned
                       03   13  0389   822           BEQL    90$                                ; xfer if not returning length
                 65    54   D0  038B   823           MOVL    R4,(R5)                            ; else save ACE length
    63    52    00    61    54   2C  038E   824 90$:      MOVC5   R4,(R1),#0,R2,(R3)                 ; copy matching ACE
                 50    56   D0  0394   8'5 100$:     MOVL    R6,R0                              ; Restore saved status
                    36 50   E8  0397   826           BLBS    R0,BYPASS_CHECK                    ; done if access granted
                                039A   827
                                039A   828 ; Processing of the protection mask depends upon what happened with the ACL
                       039A   829 ; processing.  Matching an ACE overrides group and world access.  If no
                       039A   830 ; protection mask was supplied (coded as an ORB$L_OWNER equal to zero),
                       039A   831 ; access is granted if there was no ACL, and denied if there was one.
                       039A   832
         55    5E  D0  039A   833              MOVL     SP,R5                             ; Set modified prot mask addr
         50    24  3C  039D   834              MOVZWL   #SS$_NOPRIV,R0                    ; Failure if no prot mask
               0E  11  03A0   835              BRB      120$                              ; go do SOGW check
                       03A2   836
                       03A2   837              ASSUME   ORB$L_SYS_PROT EQ ORB$W_PROT
         55 18 A9  9E  03A2   838 110$:        MOVAB    ORB$L_SYS_PROT(R9),R5             ; Set protection mask address
         50    24  3C  03A6   839              MOVZWL   #SS$_NOPRIV,R0                    ; Failure if no prot mask & ACL pres
      03 57    00  E0  03A9   840              BBS      #CHKPRO_V_ACL_PRESENT,R7,120$     ; Xfer if no ACL present
         50    01  3C  03AD   841              MOVZWL   #SS$_NORMAL,R0                    ; Else success if no prot mask & no
                       03B0   842
                       03B0   843 ; R5 set up above.
                       03B0   844
         52    68  9E  03B0   845 120$:        MOVAB    ARB$Q_PRIV(R8),R2                 ; Set up input parameters
         53    6A  D0  03B3   846              MOVL     CHPCT[$L_ACCESS(R10),R3           ; Get the desired access
         54 20 A8  9E  03B6   847              MOVAB    ARB$L_RIGHTSLIST(R8),R4           ; Set rights list descr addr
         56    69  D0  03BA   848              MOVL     ORB$L_OWNER(R9),R6                ; was there an owner?
               11  13  03BD   849              BEQL     BYPASS_CHECK                      ; xfer if not, no SOGW check
   05 0B A9    00  E0  03BF   850              BBS      #ORB$V_PROT_16,ORB$B_FLAGS(R9),130$       ; else check for full vector
             0278  30  03C4   851              BSBW     EXE$CHECKPROT                     ; do check with full vector
               03  11  03C7   852              BRB      140$                              ; go finish this check
             0247  30  03C9   853 130$:        BSBW     EXE$CHECKPROT_16                  ; do check with word value
      24 BB    51  C8  03CC   854 140$:        BISL2    R1,@CHPRET$L_PRIVS_USED(R11)      ; note any privileges used
                       03D0   855
                       03D0   856 ; At this point, the status will be set according to the protection checks
                       03D0   857 ; applied.  Now check for any overriding privileges.
                       03D0   858
                       03D0   859 BYPASS_CHECK:
         5E    10  C0  03D0   860              ADDL2    #16,SP                            ; Clean off protection vector
            0E 50  E8  03D3   861              BLBS     R0,10$                            ; xfer is successful
         52    68  9E  03D6   862              MOVAB    ARB$Q_PRIV(R8),R2                 ; Else set up input parameters
         53 04 AA  D0  03D9   863              MOVL     CHPCT[$L_FLAGS(R10),R3
             039F  30  03DD   864              BSBW     EXE$CHECK_BYPASS                  ; check for BYPASS or READALL
      24 BB    51  C8  03E0   865              BISL2    R1,@CHPRET$L_PRIVS_USED(R11)      ; note any privileges used
                       03E4   866
                       03E4   867 ; Return any security audit or alarm names from the ACL segments supplied.
                       03E4   868
                       03E4   869              ASSUME   CHPRET$W_AUDITLEN EQ CHPRET$L_AUDIT-4
                       03E4   870              ASSUME   CHPRET$L_AUDITRET EQ CHPRET$L_AUDIT+4
                       03E4   871              ASSUME   CHPRET$W_ALARMLEN EQ CHPRET$L_ALARM-4
                       03E4   872              ASSUME   CHPRET$L_ALARMRET EQ CHPRET$L_ALARM+4
                       03E4   873
         56    50  D0  03E4   874 10$:         MOVL     R0,R6                             ; save the final status
      22 57    00  E1  03E7   875              BBC      #CHKPRO_V_ACL_PRESENT,R7,RETURN_STATUS  ; if no ACL, go finish up
         54    6B  9E  03EB   876              MOVAB    CHPRET$Q_AUDITLEN(R11),R4         ; set descriptor address
               64  B5  03EE   877              TSTW     (R4)                              ; want audit journal name?
               08  13  03F0   878              BEQL     20$                               ; xfer if not, try alarm journal
         53    05  D0  03F2   879              MOVL     #ACE$C_AUDIT,R3                   ; else set the ACE type to get
               4D  10  03F5   880              BSBB     EXE$GET_AUDIT                     ; go get the journal name, if one
            10 50  E9  03F7   881              BLBC     R0,30$                            ; xfer if any errors
         54 0C AB  9E  03FA   882 20$:         MOVAB    CHPRET$W_ALARMLEN(R11),R4         ; set descriptor address
               64  B5  03FE   883              TSTW     (R4)                              ; want alarm journal name?
               08  13  0400   884              BEQL     30$                               ; xfer if not, we're done
         53    06  D0  0402   885              MOVL     #ACE$C_ALARM,R3                   ; else set the ACE type to get
```

```
                      3D   10  0405   886          BSBB    EXE$GET_AUDIT                          ; go get the journal name, if one
                   03 50   E8  0407   887          BLBS    R0,RETURN_STATUS                       ; xfer if no errors
               56     50   D0  040A   888 30$:     MOVL    R0,R6                                  ; Else change saved status
                               040D   889
                               040D   890 ; Done at last!!  Release ACL mutex, if necessary, and do the final cleanup.
                               040D   891
                               040D   892 RETURN_STATUS:
         14 0B A9     01   E1  040D   893          BBC     #ORB$V_ACL_QUEUE,ORB$B_FLAGS(R9),10$    ; Xfer if not a queue
      54   00000000'9F     D0  0412   894          MOVL    @#SCH$GL_CURPCB,R4                     ; Else get current PCB address
            50     04 A9   9E  0419   895          MOVAB   ORB$L_ACL_MUTEX(R9),R0                 ; Set mutex address
           00000000'9F     16  041D   896          JSB     @#SCH$UNLOCK                           ; Unlock mutex
                               0423   897          ENBINT                                         ; Restore IPL
               50     56   D0  0426   898 10$:     MOVL    R6,R0                                  ; Restore the final status
               0C 57  01   E1  0429   899          BBC     #CHKPRO_V_INTERNAL,R7,30$              ; xfer if system service return
               03 57  02   E1  042D   900          BBC     #CHKPRO_V_NO_CHPRET,R7,20$             ; xfer if no cleanup of CHPRET block
               5E     2C   C0  0431   901          ADDL2   #CHPRET$C_LENGTH+4,SP                  ; else remove the local CHPRET
             0FFE 8F       BA  0434   902 20$:     POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; restore work regs
                           05  0438   903          RSB                                            ; return to caller
                               0439   904
               5B     8E   D0  0439   905 30$:     MOVL    (SP)+,R11                              ; Restore local storage block addres
                               043C   906 RETURN_P1_BLOCK:
   00000000'EF        6B   0E  043C   907          INSQUE  (R11),CTL$GL_KRPFL                     ; Return block to lookaside list
                           04  0443   908          RET                                            ; return for system service entry
```

```
                          0444            .SBTTL  EXE$GET_AUDIT - SEARCH FOR SECURITY AUDIT ACE IN THE ACL
                          0444
                          0444       ;++
                          0444       ;
                          0444       ; FUNCTIONAL DESCRIPTION:
                          0444       ;
                          0444       ;        This routine searches the access control lists in the item
                          0444       ;        list for security audit or alarm entries of the specified
                          0444       ;        type.
                          0444       ;
                          0444       ; CALLING SEQUENCE:
                          0444       ;        JSB EXE$GETAUDIT
                          0444       ;
                          0444       ; INPUT PARAMETERS:
                          0444       ;        TYPE    (R3): ACE type code of audit or alarm to find
                          0444       ;        STATUS  (R6): status of the protection check
                          0444       ;        ORB     (R9): address of the object's rights block
                          0444       ;        CHPCTL  (R10): address of the protection check control block
                          0444       ;
                          0444       ; IMPLICIT INPUTS:
                          0444       ;        NONE
                          0444       ;
                          0444       ; OUTPUT PARAMETERS:
                          0444       ;        ITEM  (R4): address of item descriptor to which to write
                          0444       ;
                          0444       ; IMPLICIT OUTPUTS:
                          0444       ;        none
                          0444       ;
                          0444       ; ROUTINE VALUE:
                          0444       ;        SS$_NORMAL if ACL ok - audit found or not
                          0444       ;        SS$_IVACL if invalid
                          0444       ;
                          0444       ; SIDE EFFECTS:
                          0444       ;        NONE
                          0444       ;
                          0444       ;--
                          0444
                          0444
                          0444       EXE$GET_AUDIT:
          01FC 8F     BB  0444               PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>       ; save work registers
                 51   D4  0448               CLRL    R1                              ; start with the first ACE
     1C 0B A9    01   E1  044A               BBC     #ORB$V_ACL_QUEUE,ORB$B_FLAGS(R9),20$    ; xfer if not a queue
                          044F
                          044F       ; Handle the ACL segment queue here.
                          044F
        57    28 A9   9E  044F               MOVAB   ORB$L_ACLFL(R9),R7              ; else set address of queue head
           58    57   D0  0453               MOVL    R7,R8                           ; copy address for later
           57    67   D0  0456       10$:    MOVL    (R7),R7                         ; get address of next segment
           58    57   D1  0459               CMPL    R7,R8                           ; end of the line?
                 2F   13  045C               BEQL    50$                             ; xfer if so
                 55   D4  045E               CLRL    R5                              ; else preset segment size
   55    08 A7   0C   A3  0460               SUBW3   #ACL$C_LENGTH,ACL$W_SIZE(R7),R5 ; set segment size
        56    0C A7   9E  0465               MOVAB   ACL$L_LIST(R7),R6               ; set address of ACEs
                 0D   11  0469               BRB     40$                             ; go do the ACL check
                          046B
                          046B       ; Handle the descriptor vector here.
                          046B
```

```
              57    28 A9   D0  046B  20$:    MOVL    ORB$L_ACL_COUNT(R9),R7          ; get the number of descriptors
                       1C   13  046F          BEQL    50$                             ; xfer if no ACL supplied
              58    2C A9   D0  0471          MOVL    ORB$L_ACL_DESC(R9),R8           ; get address of descriptor list
                 55    88   D0  0475  30$:    MOVL    (R8)+,R5                        ; get a descriptor
                     0144   30  0478  40$:    BSBW    EXE$FINDACL                     ; locate the specified type
                 11    50   E8  047B          BLBS    R0,60$                          ; xfer if in this one
            09D8 8F    50   B1  047E          CMPW    R0,#SS$_NOENTRY                 ; check for normal termination
                       4E   12  0483          BNEQ    110$                            ; exit if error
           CC 0B A9    01   E0  0485          BBS     #ORB$V_ACL_QUEUE,ORB$B_FLAGS(R9),10$    ; if a queue, go get next
                    E8 57   F5  048A          SOBGTR  R7,30$                          ; else continue with next segment
                       3A   11  048D  50$:    BRB     95$                             ; Go finish up
                                048F
                                048F  ; An ACE has been found of the desired type.  Check to see if the success/failure
                                048F  ; status matches, and also that the access matches.
                                048F
                 50    61   9A  048F  60$:    MOVZBL  ACE$B_SIZE(R1),R0               ; get ACE size
                 50    08   C2  0492          SUBL    #ACE$T_AUDITNAME,R0             ; compute audit name length
                 01    50   D1  0495          CMPL    R0,#1                           ; check for minimum size
                       34   19  0498          BLSS    100$                            ; must have at least 1 byte of name
                                049A
                                049A  ; The following instruction depends on the (number and order of) registers
                                049A  ; saved upon entering EXE$GET_AUDIT.
                                049A
               07 10 AE     E9  049A          BLBC    16(SP),70$                      ; xfer if final status is failure
                                049E
                                049E  ; Verify that the success/failure status of the protection check matches the
                                049E  ; flags in the ACE.
                                049E
           0E 02 A1    00   E0  049E          BBS     #ACE$V_SUCCESS,ACE$W_FLAGS(R1),80$      ; Xfer if success matches
                       05   11  04A3          BRB     75$                             ; Else go check next segment
           07 02 A1    01   E0  04A5  70$:    BBS     #ACE$V_FAILURE,ACE$W_FLAGS(R1),80$      ; Xfer if failure matches
           A7 0B A9    01   E0  04AA  75$:    BBS     #ORB$V_ACL_QUEUE,ORB$B_FLAGS(R9),10$    ; Else xfer if a queue
                       BA   11  04AF          BRB     20$                             ; Else must be descr list
                                04B1
                                04B1  ; Now verify that the requested access is in fact enabled in the ACE.
                                04B1
              04 A1    6A   D3  04B1  80$:    BITL    CHPCTL$L_ACCESS(R10),ACE$L_ACCESS(R1)   ; for desired access?
                       C1   13  04B5          BEQL    40$                             ; xfer if not, try another ACE
                 52    84   7D  04B7          MOVQ    (R4)+,R2                        ; get descriptor
                 55    84   D0  04BA          MOVL    (R4)+,R5                        ; get return length address
                       03   13  04BD          BEQL    90$                             ; xfer if return length not needed
                 65    50   B0  04BF          MOVW    R0,(R5)                         ; else save it
63    52    00    08 A1    50   2C  04C2  90$:    MOVC5   R0,ACE$T_AUDITNAME(R1),#0,R2,(R3) ; copy the journal name
                 50    01   D0  04C9  95$:    MOVL    #SS$_NORMAL,R0                  ; set success return
                       05   11  04CC          BRB     110$                            ; go finish up
                                04CE
        50    21E4 8F       3C  04CE  100$:   MOVZWL  #SS$_IVACL,R0                   ; invalid ACL - set error
              01FC 8F       BA  04D3  110$:   POPR    #^M<R2,R3,R4,R5,R6,R7,R8>       ; save work registers
                            05  04D7          RSB
```

```
                          04D8  1017          .SBTTL  EXE$CHECKACL - CHECK FOR AN ACE IN AN ACL
                          04D8  1018
                          04D8  1019 ;++
                          04D8  1020 ;
                          04D8  1021 ; FUNCTIONAL DESCRIPTION:
                          04D8  1022 ;
                          04D8  1023 ;        This routine searches the specified access control list for an entry
                          04D8  1024 ;        that matches the specified rights list. If an entry is found, it
                          04D8  1025 ;        checks whether the entry grants the requested rights.
                          04D8  1026 ;
                          04D8  1027 ; CALLING SEQUENCE:
                          04D8  1028 ;        JSB EXE$CHECKACL
                          04D8  1029 ;
                          04D8  1030 ; INPUT PARAMETERS:
                          04D8  1031 ;        ACCESS          (R3): bitmask of access requested
                          04D8  1032 ;        RIGHTSDESC      (R4): address of rights list descriptors
                          04D8  1033 ;        ACL_LENGTH      (R5): length of ACL segment
                          04D8  1034 ;        ACL             (R6): address of ACL segment
                          04D8  1035 ;
                          04D8  1036 ; IMPLICIT INPUTS:
                          04D8  1037 ;        NONE
                          04D8  1038 ;
                          04D8  1039 ; OUTPUT PARAMETERS:
                          04D8  1040 ;        ACE             (R1): address of ACL entry matched
                          04D8  1041 ;
                          04D8  1042 ; IMPLICIT OUTPUTS:
                          04D8  1043 ;        NONE
                          04D8  1044 ;
                          04D8  1045 ; ROUTINE VALUE:
                          04D8  1046 ;        SS$_NORMAL if matching ACE found and access is granted
                          04D8  1047 ;        SS$_NOPRIV if matching ACE found and access is denied
                          04D8  1048 ;        SS$_NOENTRY if no matching ACE is found
                          04D8  1049 ;        SS$_IVACL if the ACL structure is invalid
                          04D8  1050 ;
                          04D8  1051 ; SIDE EFFECTS:
                          04D8  1052 ;        NONE
                          04D8  1053 ;
                          04D8  1054 ;--
                          04D8  1055
                          04D8  1056          .ENABLE LSB
                          04D8  1057
                          04D8  1058 EXE$CHECKACL::
        03E4 8F     BB    04D8  1059          PUSHR   #^M<R2,R5,R6,R7,R8,R9>  ; save work regs
        59   56     D0    04DC  1060          MOVL    R6,R9                   ; set address of the first ACE
        56   55     C0    04DF  1061          ADDL2   R5,R6                   ; calc end of the ACL segment
        57   59     D0    04E2  1062 10$:     MOVL    R9,R7                   ; position to next ACE
        56   57     D1    04E5  1063          CMPL    R7,R6                   ; more to go in this segment?
             59     1E    04E8  1064          BGEQU   50$                     ; xfer if not
        50   67     9A    04EA  1065          MOVZBL  ACE$B_SIZE(R7),R0       ; get the size of the current ACE
             54     13    04ED  1066          BEQL    50$                     ; xfer if at the end of the segment
        04   50     D1    04EF  1067          CMPL    R0,#4                   ; check minimum ACE size
             48     1F    04F2  1068          BLSSU   40$                     ; too small - error
   59   57   50     C1    04F4  1069          ADDL3   R0,R7,R9                ; calc the end of the current ACE
        56   59     D1    04F8  1070          CMPL    R9,R6                   ; beyond the end of the ACL segment?
             3F     1A    04FB  1071          BGTRU   40$                     ; xfer if so, note error
   E0 02 A7  08     E0    04FD  1072          BBS     #ACE$V_DEFAULT,ACE$W_FLAGS(R7),10$      ; ignore default ACEs
   01   01 A7       91    0502  1073          CMPB    ACE$B_TYPE(R7),#ACE$C_KEYID     ; else right type of ACE?
```

```
                    DA   12  0506           BNEQ    10$                           ; xfer if not, go try the next ACE
                             0508
                             0508           ASSUME  ACE$V_RESERVED EQ 0
                             0508
                    50   D4  0508           CLRL    R0                            ; pre-clear high order part
  50    02 A7    F0 8F   8B  050A           BICB3   #-<1@ACE$S_RESERVED>,ACE$W_FLAGS(R7),R0 ; get size of reserved area
     58   08 A740        DE  0510           MOVAL   ACE$L_KEY(R7)[R0],R8          ; calc start of actual keys
              59    58   D1  0515           CMPL    R8,R9                         ; check for non-null identifier list
                    22   1E  0518           BGEQU   40$                           ; branch if null - bad ACE
              52    88   D0  051A     20$:  MOVL    (R8)+,R2                      ; get an identifier from the ACE
                    2B   10  051D           BSBB    EXE$SEARCH_RIGHT              ; see if the identifier is present
                 C0 50   E9  051F           BLBC    R0,10$                        ; xfer if it is not
              59    58   D1  0522           CMPL    R8,R9                         ; at the end of the ACE?
                    F3   1F  0525           BLSSU   20$                           ; loop if not
                             0527    ;
                             0527    ; At this point an ACE has been found whose identifiers are contained in the
                             0527    ; rights lists.  Check for the desired access.
                             0527    ;
              50    24   D0  0527     30$:  MOVL    #SS$_NOPRIV,R0                ; preset status
     52    53    04 A7   CB  052A           BICL3   ACE$L_ACCESS(R7),R3,R2        ; check for access
                    03   12  052F           BNEQ    CHKACL_RETURN                 ; xfer if denied
              50    01   D0  0531           MOVL    #SS$_NORMAL,R0                ; else note successful
                             0534    CHKACL_RETURN:
              51    57   D0  0534           MOVL    R7,R1                         ; return matched ACE
               03E4 8F   BA  0537           POPR    #^M<R2,R5,R6,R7,R8,R9>        ; restore work regs
                         05  053B           RSB
                             053C    ;
                             053C    ; The ACL or an ACE within the ACL has been found to be invalid.  Note the
                             053C    ; error for the caller.
                             053C    ;
         50    21E4 8F   3C  053C     40$:  MOVZWL  #SS$_IVACL,R0                 ; else set error
                    F1   11  0541           BRB     CHKACL_RETURN                 ; go finish up
                             0543    ;
                             0543    ; The ACL does not contain the specified ACE.  Note this condition.
                             0543    ;
         50    09D8 8F   3C  0543     50$:  MOVZWL  #SS$_NOENTRY,R0               ; set error status
                    EA   11  0548           BRB     CHKACL_RETURN                 ; go finish up
                             054A
                             054A           .DISABLE LSB
```

```
                      054A           .SBTTL  EXE$SEARCH_RIGHT - SEARCH RIGHTS DESCRIPTOR FOR AN IDENTIFIER
                      054A
                      054A     ;++
                      054A     ;
                      054A     ; FUNCTIONAL DESCRIPTION:
                      054A     ;
                      054A     ;         This routine searches the specified rights segment for the given
                      054A     ;         identifier.
                      054A     ;
                      054A     ; CALLING SEQUENCE:
                      054A     ;         JSB     EXE$SEARCH_RIGHT
                      054A     ;
                      054A     ; INPUT PARAMETERS:
                      054A     ;         IDENTIFIER      (R2): identifier being sought
                      054A     ;         RIGHTSDESC      (R4): address of the rights segment descriptors
                      054A     ;
                      054A     ; IMPLICIT INPUTS:
                      054A     ;         NONE
                      054A     ;
                      054A     ; OUTPUT PARAMETERS:
                      054A     ;         ID_ADDRESS      (R1): address of the ID quadword if found
                      054A     ;         DESC_ADDRESS    (R5): address of the rights segment containing the ID
                      054A     ;
                      054A     ; IMPLICIT OUTPUTS:
                      054A     ;         NONE
                      054A     ;
                      054A     ; ROUTINE VALUE:
                      054A     ;         SS$_NORMAL if ID was found
                      054A     ;         SS$_NOSUCHID if the ID was not found
                      054A     ;
                      054A     ; SIDE EFFECTS:
                      054A     ;         NONE
                      054A     ;
                      054A     ;--
                      054A
                      054A           ASSUME  UIC$K_UIC_FORMAT EQ 0
                      054A           ASSUME  UIC$K_ID_FORMAT EQ 2
                      054A           ASSUME  UIC$V_FORMAT EQ 30
                      054A           ASSUME  UIC$K_MATCH_ALL EQ -1
                      054A
                      054A     EXE$SEARCH_RIGHT::
           5A    DD   054A           PUSHL   R10                        ; save work registers
           54    DD   054C           PUSHL   R4
           53    DD   054E           PUSHL   R3
     5A    52    D2   0550           MCOML   R2,R10                     ; see if match-all specified
           07    12   0553           BNEQ    5$                         ; branch if not
     5A    52    D0   0555           MOVL    R2,R10                     ; set test mask to all ones
           52    D4   0558           CLRL    R2                         ; search pattern is zero
           27    11   055A           BRB     30$                        ; and execute match code
                      055C
  4B 52    1E    E0   055C     5$:   BBS     #30,R2,50$                 ; xfer if invalid identifier format
           5A    D4   0560           CLRL    R10                        ; preset UIC mask
           52    D5   0562           TSTL    R2                         ; check for a UIC type identifier
           1D    19   0564           BLSS    30$                        ; xfer if not a UIC
                      0566     ;
                      0566     ; Form a wildcard mask based upon the UIC entry in the ACE.
                      0566     ;
```

```
SYS
Sym
IND
IPL
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
ITE
LOC
LOC
LOC
LOC
LOC
MAX
MAX
MAX
MIN
NEX
NEX
OFF
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
ORB
PCB
```

```
    00003FFF 8F   52   0E    10   ED  0566        CMPZV   #UIC$V_GROUP,#UIC$S_GROUP,R2,#UIC$K_WILD_GROUP
                                      056F                                                ; wildcard group?
                             05   12  056F        BNEQ    10$                             ; xfer if not
                        5A   52   D0  0571        MOVL    R2,R10                          ; get the UIC with wild group
                             5A   B4  0574        CLRW    R10                             ; zap the member for now
                   FFFF 8F   52   B1  0576  10$:  CMPW    R2,#UIC$K_WILD_MEMBER           ; wildcard member?
                             03   12  057B        BNEQ    20$                             ; xfer if not
                        5A   01   AE  057D        MNEGW   #1,R10                          ; else note it
                        52   5A   CA  0580  20$:  BICL    R10,R2                          ; mask out unneeded portions
                                      0583  ;
                                      0583  ; At this point an identifier exists in R2.  Now scan the rights list segments
                                      0583  ; to see if it exists within the rights lists.
                                      0583  ;
                        55   84   D0  0583  30$:  MOVL    (R4)+,R5                        ; else get address of a descriptor
                             23   13  0586        BEQL    50$                             ; xfer if at the end...ID not found
                        53   65   3C  0588        MOVZWL  (R5),R3                         ; else get size of descriptor
             53   53   FD 8F   78  058B        ASHL    #-3,R3,R3                       ; get number of entries
                             F1   13  0590        BEQL    30$                             ; xfer if none to check
                   51   04 A5   D0  0592        MOVL    4(R5),R1                        ; get starting address
                        50   61   D0  0596  40$:  MOVL    (R1),R0                         ; get the identifier
                             E8   13  0599        BEQL    30$                             ; xfer if no more
                        50   5A   CA  059B        BICL    R10,R0                          ; mask out any unneeded portions
                        50   52   D1  059E        CMPL    R2,R0                           ; ACE & rights list identifier match?
                             0F   13  05A1        BEQL    60$                             ; xfer if so, next identifier please
                        51   08   C0  05A3        ADDL    #ARB$S_RIGHTSDESC,R1            ; point to next identifier
                          ED 53   F5  05A6        SOBGTR  R3,40$                          ; go try it
                             D8   11  05A9        BRB     30$                             ; if exhausted, try next rights list
                                      05AB
                   50   21EC 8F   3C  05AB  50$:  MOVZWL  #SS$_NOSUCHID,R0                ; set status
                             03   11  05B0        BRB     70$                             ; go finish up
                                      05B2
                        50   01   D0  05B2  60$:  MOVL    #SS$_NORMAL,R0                  ; set status
                        53   8E   D0  05B5  70$:  MOVL    (SP)+,R3                        ; restore work registers
                        54   8E   D0  05B8        MOVL    (SP)+,R4
                        5A   8E   D0  05BB        MOVL    (SP)+,R10
                                  05  05BE        RSB                                     ; return to caller
```

```
                     05BF          .SBTTL  EXE$FINDACL - SEARCH FOR A PARTICULAR ACE IN THE ACL
                     05BF
                     05BF    ;++
                     05BF    ;
                     05BF    ; FUNCTIONAL DESCRIPTION:
                     05BF    ;
                     05BF    ;       This routine searches the specified ACL segment for an entry
                     05BF    ;       of the specified type.
                     05BF    ;
                     05BF    ; CALLING SEQUENCE:
                     05BF    ;       JSB EXE$FINDACL
                     05BF    ;
                     05BF    ; INPUT PARAMETERS:
                     05BF    ;       TYPE            (R3): type code of ACE to find
                     05BF    ;       ACL_LENGTH      (R5): length of ACL segment
                     05BF    ;       ACL             (R6): address of ACL segment
                     05BF    ;       PREV_ACE        (R1): address of previously found ACE
                     05BF    ;
                     05BF    ; IMPLICIT INPUTS:
                     05BF    ;       NONE
                     05BF    ;
                     05BF    ; OUTPUT PARAMETERS:
                     05BF    ;       ACE             (R1): address of found entry
                     05BF    ;
                     05BF    ; IMPLICIT OUTPUTS:
                     05BF    ;       NONE
                     05BF    ;
                     05BF    ; ROUTINE VALUE:
                     05BF    ;       SS$_NORMAL if entry found
                     05BF    ;       SS$_NOENTRY if entry not found
                     05BF    ;       SS$_IVACL if ACL format is invalid
                     05BF    ;
                     05BF    ; SIDE EFFECTS:
                     05BF    ;       NONE
                     05BF    ;
                     05BF    ;--
                     05BF
                     05BF    EXE$FINDACL::
           58    DD  05BF            PUSHL   R8                              ; save work regs
           57    DD  05C1            PUSHL   R7
  57   56  55    C1  05C3            ADDL3   R5,R6,R7                        ; calc the end of the ACL segment
           51    D5  05C7            TSTL    R1                              ; any previous entry?
           0A    13  05C9            BEQL    10$                             ; branch if not
       50  61    9A  05CB            MOVZBL  ACE$B_SIZE(R1),R0               ; else get size of ACE
           32    13  05CE            BEQL    40$                             ; xfer if at the end of the segment
       51  50    C0  05D0            ADDL    R0,R1                           ;   else point to the next one
           03    11  05D3            BRB     20$
                     05D5
       51  56    D0  05D5    10$:    MOVL    R6,R1                           ; set up for the first ACE
       57  51    D1  05D8    20$:    CMPL    R1,R7                           ; at the end of the ACL?
           25    1E  05DB            BGEQU   40$                             ; xfer if so, done for the moment
       50  61    9A  05DD            MOVZBL  ACE$B_SIZE(R1),R0               ; else get size of ACE
           20    13  05E0            BEQL    40$                             , Xfer if at the end of the segment
       04  50    D1  05E2            CMPL    R0,#4                           ; check minimum ACE size
           14    1F  05E5            BLSSU   30$                             ; too small - error
  58   51  50    C1  05E7            ADDL3   R0,R1,R8                        ;   and point to the next one
       57  58    D1  05EB            CMPL    R8,R7                           ; check end of ACE against ACL
```

```
            0B   1A  05EE  1265          BGTRU   30$                     ; xfer if out of range
    53   01 A1   91  05F0  1266          CMPB    ACE$B_TYPE(R1),R3       ; found desired type?
            13   13  05F4  1267          BEQL    50$                     ; xfer if so, time to go
       51   58   D0  05F6  1268          MOVL    R8,R1                   ; advance to next ACE
            DD   11  05F9  1269          BRB     20$                     ; go test for the end
                     05FB  1270
   50    21E4 8F   3C  05FB  1271 30$:     MOVZWL  #SS$_IVACL,R0           ; else set error status
            0A   11  0600  1272          BRB     60$                     ; go finish up
                     0602  1273
   50    09D8 8F   3C  0602  1274 40$:     MOVZWL  #SS$_NOENTRY,R0         ; no entry found
            03   11  0607  1275          BRB     60$                     ; go finish up
                     0609  1276
       50   01   D0  0609  1277 50$:     MOVL    #SS$_NORMAL,R0          ; entry found
       57   8E   D0  060C  1278 60$:     MOVL    (SP)+,R7                ; restore work regs
       58   8E   D0  060F  1279          MOVL    (SP)+,R8
                 05  0612  1280          RSB
```

```
                           0613          .SBTTL  EXE$CHECKPROT_16 - DO STANDARD SOGW CHECK WITH WORD INPUT
                           0613
                           0613  ;++
                           0613  ;
                           0613  ; FUNCTIONAL DESCRIPTION:
                           0613  ;
                           0613  ;       This routine performs the standard "system - owner - group -
                           0613  ;       world" protection check using the information supplied.  This
                           0613  ;       routine differs from EXE$CHKPROT in that it expects a pointer
                           0613  ;       to a word protection mask input instead of a pointer to a
                           0613  ;       longword array.
                           0613  ;
                           0613  ; CALLING SEQUENCE:
                           0613  ;       JSB EXE$CHECKPROT
                           0613  ;
                           0613  ; INPUT PARAMETERS:
                           0613  ;       PRIV_MASK       (R2): address of accessor privilege mask
                           0613  ;       ACCESS          (R3): bitmask of access requested
                           0613  ;       RIGHTSDESC      (R4): address of rights list descriptors
                           0613  ;       PROTECTION      (R5): address of the protection word to use
                           0613  ;       OWNER           (R6): owner UIC of object
                           0613  ;
                           0613  ; IMPLICIT INPUTS:
                           0613  ;       NONE
                           0613  ;
                           0613  ; OUTPUT PARAMETERS:
                           0613  ;       PRIVS_USED      (R1): bitmask of privileges used to gain access
                           0613  ;
                           0613  ; IMPLICIT OUTPUTS:
                           0613  ;       NONE
                           0613  ;
                           0613  ; ROUTINE VALUE:
                           0613  ;       SS$_NORMAL: access is granted
                           0613  ;       SS$_NOPRIV: access if denied
                           0613  ;
                           0613  ; SIDE EFFECTS:
                           0613  ;       NONE
                           0613  ;
                           0613  ;--
                           0613
                           0613  EXE$CHECKPROT_16::
                 5A    DD  0613          PUSHL   R10                     ; save work regs
                 58    DD  0615          PUSHL   R8
                 57    DD  0617          PUSHL   R7
                 53    DD  0619          PUSHL   R3
                 55    DD  061B          PUSHL   R5
  7E    65   04  0C    EF  061D          EXTZV   #12,#4,(R5),-(SP)       ; save the world protection bits
             6E  10    C8  0622          BISL2   #ARM$M_CONTROL,(SP)     ; control access denied
  7E    65   04  08    EF  0625          EXTZV   #8,#4,(R5),-(SP)        ; save the group protection bits
             6E  10    C8  062A          BISL2   #ARM$M_CONTROL,(SP)     ; control access denied
  7E    65   04  04    EF  062D          EXTZV   #4,#4,(R5),-(SP)        ; save the owner protection bits
  7E    65   04  00    EF  0632          EXTZV   #0,#4,(R5),-(SP)        ; save the system protection bits
             55  5E    D0  0637          MOVL    SP,R5                   ; save address of protection array
             7E  01    CE  063A          MNEGL   #1,-(SP)                ; indicate entry type
                 0A    11  063D          BRB     EXE$CHECKPROT_CMN       ; go join common code
```

```
                  063F          .SBTTL  EXE$CHECKPROT - DO STANDARD SOGW PROTECTION CHECK
                  063F
                  063F   ;++
                  063F   ;
                  063F   ; FUNCTIONAL DESCRIPTION:
                  063F   ;
                  063F   ;      This routine performs the standard "system - owner - group -
                  063F   ;      world" protection check using the information supplied.
                  063F   ;
                  063F   ; CALLING SEQUENCE:
                  063F   ;      JSB EXE$CHECKPROT
                  063F   ;
                  063F   ; INPUT PARAMETERS:
                  063F   ;      PRIV_MASK       (R2): address of accessor privilege mask
                  063F   ;      ACCESS          (R3): bitmask of access requested
                  063F   ;      RIGHTSDESC      (R4): address of rights list descriptors
                  063F   ;      PROTECTION      (R5): address of protection mask
                  063F   ;      OWNER           (R6): owner UIC of object
                  063F   ;
                  063F   ; IMPLICIT INPUTS:
                  063F   ;      NONE
                  063F   ;
                  063F   ; OUTPUT PARAMETERS:
                  063F   ;      PRIVS_USED      (R1): bitmask of privileges used to gain access
                  063F   ;
                  063F   ; IMPLICIT OUTPUTS:
                  063F   ;      NONE
                  063F   ;
                  063F   ; ROUTINE VALUE:
                  063F   ;      SS$_NORMAL: access is granted
                  063F   ;      SS$_NOPRIV: access if denied
                  063F   ;
                  063F   ; SIDE EFFECTS:
                  063F   ;      NONE
                  063F   ;
                  063F   ;--
                  063F
                  063F   EXE$CHECKPROT::
         5A   DD  063F          PUSHL   R10                     ; save work regs
         58   DD  0641          PUSHL   R8
         57   DD  0643          PUSHL   R7
         53   DD  0645          PUSHL   R3
         7E   D4  0647          CLRL    -(SP)                   ; indicate entry type
                  0649
                  0649   EXE$CHECKPROT_CMN:
    50   01   D0  0649          MOVL    #SS$_NORMAL,R0          ; assume success
         51   D4  064C          CLRL    R1                      ; no privs used yet
    5A   64   D0  064E          MOVL    (R4),R10                ; get address of first descriptor
5A   04 BA   D0  0651          MOVL    @4(R10),R10             ; get the UIC from first descriptor
                  0655   ;
                  0655   ; Check for owner access first since it will be the most common
                  0655   ;
    56   5A   D1  0655          CMPL    R10,R6                  ; UICs match?
         09   12  0658          BNEQ    10$                     ; xfer if not, on to the next test
57   04 A5   D2  065A          MCOML   4(R5),R7                ; get access bits
    53   57   CA  065E          BICL    R7,R3                   ; see if access allowed
         56   13  0661          BEQL    50$                     ; xfer if it is
```

```
                                  0663  ;
                                  0663  ; Try world access next.
                                  0663  ;
             57    0C A5     D2   0663  10$:    MCOML   12(R5),R7                  ; get access bits
                53    57     CA   0667          BICL    R7,R3                      ; see if access allowed
                      4D     13   066A          BEQL    50$                        ; xfer if so
                                  066C  ;
                                  066C  ; Since world access failed, try group access next
                                  066C  ;
     5A    5A    F0 8F       9C   066C          ROTL    #-16,R10,R10               ; get accessor group in low word
     58    56    F0 8F       9C   0671          ROTL    #-16,R6,R8                 ; get owner group in low word
                58    5A     B1   0676          CMPW    R10,R8                     ; check for group number match
                      21     12   0679          BNEQ    20$                        ; xfer if not a match
           5A    C000 8F     B3   067B          BITW    #^XC000,R10                ; check if UIC format accessor
                      1A     12   0680          BNEQ    20$                        ; branch if not - no group
             57    08 A5     D2   0682          MCOML   8(R5),R7                   ; get access bits
                53    57     CA   0686          BICL    R7,R3                      ; see if access allowed
                      2E     13   0689          BEQL    50$                        ; xfer if allowed
                                  068B  ;
                                  068B  ; Try for group access via the system protection field and GRPPRV
                                  068B  ;
             0D 62    22     E1   068B          BBC     #PRV$V_GRPPRV,(R2),20$     ; branch if no GRPPRV
                57    65     D2   068F          MCOML   (R5),R7                    ; get access bits
                53    57     CA   0692          BICL    R7,R3                      ; see if access allowed
                      1F     12   0695          BNEQ    40$                        ; xfer if not allowed
                51    10     C8   0697          BISL    #CHP$M_GRPPRV,R1           ; else note GRPPRV used
                      1D     11   069A          BRB     50$                        ; go finish up
                                  069C  ;
                                  069C  ; Finally check the system protection field
                                  069C  ;
             09 62    1C     E0   069C  20$:    BBS     #PRV$V_SYSPRV,(R2),30$     ; branch if no SYSPRV
     00000000'9F      5A     B1   06A0          CMPW    R10,@#EXE$GL_SYSUIC        ; system group?
                      0D     1A   06A7          BGTRU   40$                        ; xfer if not
                57    65     D2   06A9  30$:    MCOML   (R5),R7                    ; get access bits
                53    57     CA   06AC          BICL    R7,R3                      ; see if access allowed
                      05     12   06AF          BNEQ    40$                        ; xfer if not allowed
                51    01     C8   06B1          BISL    #CHP$M_SYSPRV,R1           ; else note SYSPRV used
                      03     11   06B4          BRB     50$                        ; go finish up
                                  06B6  ;
                                  06B6  ; Note that no access was allowed
                                  06B6  ;
                50    24     D0   06B6  40$:    MOVL    #SS$_NOPRIV,R0
                                  06B9  ;
                                  06B9  ; Finally, clean up the stack.
                                  06B9  ;
             06 8E           E9   06B9  50$:    BLBC    (SP)+,60$                  ; branch if normal entry
                5E    10     C0   06BC          ADDL2   #16,SP                     ; else clean up protection array
                55    8E     D0   06BF          MOVL    (SP)+,R5                   ; restore one reg
                53    8E     D0   06C2  60$:    MOVL    (SP)+,R3                   ; restore remaining work regs
                57    8E     D0   06C5          MOVL    (SP)+,R7
                58    8E     D0   06C8          MOVL    (SP)+,R8
                5A    8E     D0   06CB          MOVL    (SP)+,R10
                             05   06CE          RSB                                ;   and return
```

```
                          06CF          .SBTTL  EXE$CHECKACMODE - DO ACCESS MODE PROTECTION CHECK
                          06CF
                          06CF    ;++
                          06CF    ;
                          06CF    ; FUNCTIONAL DESCRIPTION:
                          06CF    ;
                          06CF    ;       This routine performs the access mode protection check.  The
                          06CF    ;       accessor access mode must be less than or equal to the access
                          06CF    ;       mode.  For the per-access mode protection check, this must be
                          06CF    ;       true for each field in the access mode vector for which access
                          06CF    ;       is intended.
                          06CF    ;
                          06CF    ; CALLING SEQUENCE:
                          06CF    ;       JSB     EXE$CHECKACMODE
                          06CF    ;
                          06CF    ; INPUT PARAMETERS:
                          06CF    ;       ACCESS          (R3): bitmask of intended access
                          06CF    ;       ACCESS_MODE     (R4): access mode of accessor
                          06CF    ;       MODE_PROT       (R5): access mode protection vector
                          06CF    ;
                          06CF    ; IMPLICIT INPUTS:
                          06CF    ;       NONE
                          06CF    ;
                          06CF    ; OUTPUT PARAMETERS:
                          06CF    ;       NONE
                          06CF    ;
                          06CF    ; IMPLICIT OUTPUTS:
                          06CF    ;       NONE
                          06CF    ;
                          06CF    ; ROUTINE VALUE:
                          06CF    ;       SS$_NORMAL: access granted
                          06CF    ;       SS$_NOPRIV: access denied
                          06CF    ;
                          06CF    ; SIDE EFFECTS:
                          06CF    ;       NONE
                          06CF    ;
                          06CF    ;--
                          06CF
                          06CF   EXE$CHECKACMODE::
                   59  DD 06CF          PUSHL   R9                      ; save work regs
                   58  DD 06D1          PUSHL   R8
              50   24  D0 06D3          MOVL    #SS$_NOPRIV,R0          ; assume failure
              04   55  D1 06D6          CMPL    R5,#4                   ; value or vector?
                   07  1A 06D9          BGTRU   10$                     ; xfer if a vector
                          06DB    ;
                          06DB    ; Perform a simple access mode check.
                          06DB    ;
              55   54  D1 06DB          CMPL    R4,R5                   ; else check for inner mode
                   1E  1B 06DE          BLEQU   30$                     ; xfer if so
                   1F  11 06E0          BRB     40$                     ; else note failure
                          06E2    ;
                          06E2    ; Perform the per-access mode check.
                          06E2    ;
                   59  D4 06E2   10$:   CLRL    R9                      ; reset index
         58   20   59  C3 06E4   20$:   SUBL3   R9,#32,R8               ; compute bits left to test
    59   53   58   59  EA 06E8          FFS     R9,R8,R3,R9             ; find next access bit set
                   0F  13 06ED          BEQL    30$                     ; no more bits found - done
```

```
            58    59    59   C1  06EF   1506            ADDL3   R9,R9,R8                  ; two bits at a time
      54    65    02    58   ED  06F3   1507            CMPZV   R8,#2,(R5),R4             ; accessor mode more privileged?
                        07   1F  06F8   1508            BLSSU   40$                       ; xfer if not
               E6 59    20   F2  06FA   1509            AOBLSS  #32,R9,20$                ; move to next bit and loop
                                 06FE   1510
                  50    01   D0  06FE   1511  30$:      MOVL    #SS$_NORMAL,R0            ; else set access allowed
                  58    8E   D0  0701   1512  40$:      MOVL    (SP)+,R8                  ; restore work regs
                  59    8E   D0  0704   1513            MOVL    (SP)+,R9
                             05  0707   1514            RSB                               ;   and return
```

```
                              0708  1516          .SBTTL  EXE$CHECKCLASS - DO NON-DISCRETIONARY SECURITY CHECK
                              0708  1517
                              0708  1518 ;++
                              0708  1519 ;
                              0708  1520 ; FUNCTIONAL DESCRIPTION:
                              0708  1521 ;
                              0708  1522 ;       This routine performs the non-discretionary security check, using
                              0708  1523 ;       the specified security and integrity levels and category masks.
                              0708  1524 ;
                              0708  1525 ; CALLING SEQUENCE:
                              0708  1526 ;       JSB EXE$CHECKCLASS
                              0708  1527 ;
                              0708  1528 ; INPUT PARAMETERS:
                              0708  1529 ;       PRIV_MASK       (R2): address of accessor privilege mask
                              0708  1530 ;       ACCESS          (R3): bitmask of access requested
                              0708  1531 ;                             bit 0 => read
                              0708  1532 ;                             bit 1 => write
                              0708  1533 ;       ACC_CLASS       (R4): address of accessor's classification
                              0708  1534 ;       MIN_CLASS       (R5): address of minimum classification of object
                              0708  1535 ;       MAX_CLASS       (R6): address of maximum classification of object
                              0708  1536 ;
                              0708  1537 ; IMPLICIT INPUTS:
                              0708  1538 ;       NONE
                              0708  1539 ;
                              0708  1540 ; OUTPUT PARAMETERS:
                              0708  1541 ;       PRIVS_USED      (R1): bitmask of privileges used to gain access
                              0708  1542 ;
                              0708  1543 ; IMPLICIT OUTPUTS:
                              0708  1544 ;       NONE
                              0708  1545 ;
                              0708  1546 ; ROUTINE VALUE:
                              0708  1547 ;       SS$_NORMAL if access granted
                              0708  1548 ;       SS$_NOPRIV if access denied
                              0708  1549 ;
                              0708  1550 ; SIDE EFFECTS:
                              0708  1551 ;       NONE
                              0708  1552 ;
                              0708  1553 ;--
                              0708  1554
                              0708  1555 EXE$CHECKCLASS::
                   51    D4   0708  1556         CLRL    R1                          ; no privileges used yet
                              070A  1557 ;
                              070A  1558 ; Check for read access requested.
                              070A  1559 ;
                              070A  1560         ASSUME  CHP$M_READ EQ 1
                30 53    E9   070A  1561         BLBC    R3,10$
                              070D  1562 ;
                              070D  1563 ; Check the security level using the simple securrity property.
                              070D  1564 ;
              65   64    91   070D  1565         CMPB    CLS$B_SECUR_LEV(R4),CLS$B_SECUR_LEV(R5)          ; access > min?
                   69    1F   0710  1566         BLSSU   60$                                              ; no, fail it
 50    04 A5    04 A4    CB   0712  1567         BICL3   CLS$Q_SECUR_CAT(R4),CLS$Q_SECUR_CAT(R5),R0       ; low part OK?
                   61    12   0718  1568         BNEQ    60$                                              ; xfer if not
 50    08 A5    08 A4    CB   071A  1569         BICL3   CLS$Q_SECUR_CAT+4(R4),CLS$Q_SECUR_CAT+4(R5),R0   ; high part OK?
                   59    12   0720  1570         BNEQ    60$                                   ; xfer if high part checks out
                              0722  1571 ;
                              0722  1572 ; Check the integrity level using the simple integrity property.
```

```
                          0722  1573 ;
      01 A4   01 A6   91  0722  1574          CMPB    CLS$B_INTEG_LEV(R6),CLS$B_INTEG_LEV(R4)         ; access < max?
                 52   1F  0727  1575          BLSSU   60$                                             ; no, fail it
  50  0C A4   0C A6   CB  0729  1576          BICL3   CLS$Q_INTEG_CAT(R6),CLS$Q_INTEG_CAT(R4),R0      ; low part OK?
                 4A   12  072F  1577          BNEQ    60$                                             ; xfer if not
  50  10 A4   10 A6   CB  0731  1578          BICL3   CLS$Q_INTEG_CAT+4(R6),CLS$Q_INTEG_CAT+4(R4),R0  ; high part OK?
                 42   12  0737  1579          BNEQ    60$                             ; xfer if high part does not check o
                          0739  1580 ;
                          0739  1581 ; Check for write access requested.
                          0739  1582 ;
         3A 53   01   E1  0739  1583          BBC     #CHP$V_WRITE,R3,50$             ; see if write access requested
                          073D  1584 ;
                          073D  1585 ; Check the security level using the star property.
                          073D  1586 ;
            64   66   91  073D  1587 10$:     CMPB    CLS$B_SECUR_LEV(R6),CLS$B_SECUR_LEV(R4)         ; access < max?
                 10   1F  0740  1588          BLSSU   20$                                             ; no, fail it
  50  04 A4   04 A6   CB  0742  1589          BICL3   CLS$Q_SECUR_CAT(R6),CLS$Q_SECUR_CAT(R4),R0      ; low part OK?
                 08   12  0748  1590          BNEQ    20$                                             ; xfer if not
  50  08 A4   08 A6   CB  074A  1591          BICL3   CLS$Q_SECUR_CAT+4(R6),CLS$Q_SECUR_CAT+4(R4),R0  ; high part OK?
                 07   13  0750  1592          BEQL    30$                             ; xfer if high part checks out
                          0752  1593
         25 62   21   E1  0752  1594 20$:     BBC     #PRV$V_DOWNGRADE,(R2),60$ ; branch if no DOWNGRADE
            51   08   C8  0756  1595          BISL    #CHP$M_DOWNGRADE,R1      ; else note the use
                          0759  1596 ;
                          0759  1597 ; Check the integrity level using the star property.
                          0759  1598 ;
      01 A5   01 A4   91  0759  1599 30$:     CMPB    CLS$B_INTEG_LEV(R4),CLS$B_INTEG_LEV(R5)         ; access > min?
                 10   1F  075E  1600          BLSSU   40$                                             ; no, fail it
  50  0C A5   0C A4   CB  0760  1601          BICL3   CLS$Q_INTEG_CAT(R4),CLS$Q_INTEG_CAT(R5),R0      ; low part OK?
                 08   12  0766  1602          BNEQ    40$                                             ; xfer if not
  50  10 A5   10 A4   CB  0768  1603          BICL3   CLS$Q_INTEG_CAT+4(R4),CLS$Q_INTEG_CAT+4(R5),R0  ; high part OK?
                 07   13  076E  1604          BEQL    50$                                             ; branch if OK
                          0770  1605
         07 62   20   E1  0770  1606 40$:     BBC     #PRV$V_UPGRADE,(R2),60$ ; branch if no UPGRADE
            51   04   C8  0774  1607          BISL    #CHP$M_UPGRADE,R1       ; else note the use
                          0777  1608
            50   01   D0  0777  1609 50$:     MOVL    #SS$_NORMAL,R0          ; note access granted
                      05  077A  1610          RSB
            50   24   D0  077B  1611 60$:     MOVL    #SS$_NOPRIV,R0          ; note denial of access
                      05  077E  1612          RSB
```

```
                             077F          .SBTTL  EXE$CHECK_BYPASS - CHECK FOR BYPASS OR READALL PRIVILEGES
                             077F
                             077F  ;++
                             077F  ;
                             077F  ; FUNCTIONAL DESCRIPTION:
                             077F  ;
                             077F  ;       This routine checks for either the BYPASS privilege (regardless of the
                             077F  ;       access desired) or the READALL privilege and read access.  In which
                             077F  ;       case, success is returned.  Otherwise access is denied.
                             077F  ;
                             077F  ; CALLING SEQUENCE:
                             077F  ;       JSB     EXE$CHECK_BYPASS
                             077F  ;
                             077F  ; INPUT PARAMETERS:
                             077F  ;       STATUS          (R0): protection check status so far
                             077F  ;       PRIV_MASK       (R2): address of the accessor privilege mask
                             077F  ;       ACCESS          (R3): bitmask of access requested
                             077F  ;
                             077F  ; IMPLICIT INPUTS:
                             077F  ;       NONE
                             077F  ;
                             077F  ; OUTPUT PARAMETERS:
                             077F  ;       STATUS          (R0): success or failure, depending on privs
                             077F  ;       PRIVS_USED      (R1): bitmask if privileges used to gain access
                             077F  ;
                             077F  ; IMPLICIT OUTPUTS:
                             077F  ;       NONE
                             077F  ;
                             077F  ; ROUTINE VALUE:
                             077F  ;       SS$_NORMAL: access is granted
                             077F  ;       SS$_NOPRIV: access is denied
                             077F  ;
                             077F  ; SIDE EFFECTS:
                             077F  ;       NONE
                             077F  ;
                             077F  ;--
                             077F
                             077F  EXE$CHECK_BYPASS::
                    51   D4  077F          CLRL    R1                      ; no privs used so far
               24   50   D1  0781          CMPL    R0,#SS$_NOPRIV          ; see if we are in fact checking NOPRIV
                    09   13  0784          BEQL    10$                     ; Xfer if so, see if privilege override
000021E4 8F         50   D1  0786          CMPL    R0,#SS$_IVACL           ; Else check for an invalid ACL
                    17   12  078D          BNEQ    40$                     ; Xfer if error cannot be overridden
            09 53   02   E1  078F  10$:    BBC     #CHP$V_USEREADALL,R3,20$ ; xfer if READALL not applicable
            05 62   23   E1  0793          BBC     #PRV$V_READALL,(R2),20$ ; branch if no READALL
               51   20   C8  0797          BISL2   #CHP$M_READALL,R1       ; else note READALL used
                    07   11  079A          BRB     30$                     ; successful
                             079C
            06 62   1D   E1  079C  20$:    BBC     #PRV$V_BYPASS,(R2),40$  ; branch if no BYPASS
               51   02   C8  07A0          BISL2   #CHP$M_BYPASS,R1        ; else note BYPASS used
                             07A3
               50   01   D0  07A3  30$:    MOVL    #SS$_NORMAL,R0          ; set success
                         05  07A6  40$:    RSB                             ; and return to caller
                             07A7
                             07A7          .END
```

```
ACE$B_SIZE                = 00000000    D           CHP$_FLAGS                  = 00000002    D
ACE$B_TYPE                = 00000001    D           CHP$_MATCHEDACE             = 00000011    D
ACE$C_ALARM               = 00000006    D           CHP$_MAXCLASS               = 0000000B    D
ACE$C_AUDIT               = 00000005    D           CHP$_MAX_CODE               = 00000013    D
ACE$C_KEYID               = 00000001    D           CHP$_MINCLASS               = 0000000A    D
ACE$L_ACCESS              = 00000004    D           CHP$_MODE                   = 00000008    D
ACE$L_KEY                 = 00000008    D           CHP$_MODES                  = 00000009    D
ACE$S_RESERVED            = 00000004    D           CHP$_OWNER                  = 0000000C    D
ACE$T_AUDITNAME           = 00000008    D           CHP$_PRIV                   = 00000003    D
ACE$V_DEFAULT             = 00000008    D           CHP$_PRIVUSED               = 00000012    D
ACE$V_FAILURE             = 00000001    D           CHP$_PROT                   = 0000000D    D
ACE$V_RESERVED            = 00000000    D           CHP$_RIGHTS                 = 00000006    D
ACE$V_SUCCESS             = 00000000    D           CHPCTL$B_MODE               = 00000008    D
ACE$W_FLAGS               = 00000002    D           CHPCTL$C_LENGTH             = 0000000C    D
ACL$C_LENGTH              = 0000000C    D           CHPCTL$L_ACCESS             = 00000000    D
ACL$L_LIST                = 0000000C    D           CHPCTL$L_FLAGS              = 00000004    D
ACL$W_SIZE                = 00000008    D           CHPCTL$M_READ               = 00000001    D
ACL_LIST                  = 000000EC    D           CHPCTL$M_WRITE              = 00000002    D
ARB$C_HEADER              = 00000030    D           CHPCTL_INDEX                = 00000002    D
ARB$L_RIGHTSLIST          = 00000020    D           CHPRET$C_LENGTH             = 00000028    D
ARB$Q_PRIV                = 00000000    D           CHPRET$L_ALARM              = 00000010    D
ARB$R_CLASS               = 0000000C    D           CHPRET$L_ALARMRET           = 00000014    D
ARB$S_CLASS               = 00000014    D           CHPRET$L_AUDIT              = 00000004    D
ARB$S_RIGHTSDESC          = 00000008    D           CHPRET$L_AUDITRET           = 00000008    D
ARB$S_RIGHTSLIST          = 00000010    D           CHPRET$L_MATCHED_ACE        = 0000001C    D
ARB_INDEX                 = 00000000    D           CHPRET$L_MATCHED_ACERET     = 00000020    D
ARM$M_CONTROL             = 00000010    D           CHPRET$L_PRIVS_USED         = 00000024    D
BADPARAM                    000000EA R  D   02      CHPRET$W_ALARMLEN           = 0000000C    D
BADPARAM1                   0000017C R  D   02      CHPRET$W_AUDITLEN           = 00000000    D
BUG$_KRPEMPTY               ********    X   02      CHPRET$W_MATCHED_ACELEN     = 00000018    D
BYPASS_CHECK                000003D0 R  D   02      CHPRET_INDEX                = 00000003    D
CHKACL_RETURN               00000534 R  D   02      CLS$B_INTEG_LEV             = 00000001    D
CHKPRO_ARGCOUNT           = 00000000    D           CLS$B_SECUR_LEV             = 00000000    D
CHKPRO_ITMLST             = 00000004    D           CLS$Q_INTEG_CAT             = 0000000C    D
CHKPRO_M_ACL_PRESENT      = 00000001    D           CLS$Q_SECUR_CAT             = 00000004    D
CHKPRO_M_INTERNAL         = 00000002    D           CTL$GL_KRPFL                  ********    X   02
CHKPRO_M_NO_CHPRET        = 00000004    D           DSC$C_S_BLN                 = 00000008    D
CHKPRO_V_ACL_PRESENT      = 00000000    D           EXE$CHECKACL                  000004D8 RG D   02
CHKPRO_V_INTERNAL         = 00000001    D           EXE$CHECKACMODE               000006CF RG D   02
CHKPRO_V_NO_CHPRET        = 00000002    D           EXE$CHECKCLASS                00000708 RG D   02
CHP$M_BYPASS              = 00000002    D           EXE$CHECKPROT                 0000063F RG D   02
CHP$M_DOWNGRADE           = 00000008    D           EXE$CHECKPROT_16              00000613 RG D   02
CHP$M_GRPPRV              = 00000010    D           EXE$CHECKPROT_CMN             00000649 R  D   02
CHP$M_READ                = 00000001    D           EXE$CHECK_BYPASS              0000077F RG D   02
CHP$M_READALL             = 00000020    D           EXE$CHKPRO                    00000072 RG D   02
CHP$M_SYSPRV              = 00000001    D           EXE$CHKPRO_CMN                00000282 R  D   02
CHP$M_UPGRADE             = 00000004    D           EXE$CHKPRO_INT                0000027B RG D   02
CHP$V_USEREADALL          = 00000002    D           EXE$FINDACL                   000005BF RG D   02
CHP$V_WRITE               = 00000001    D           EXE$GET_AUDIT                 00000444 R  D   02
CHP$_ACCESS               = 00000001    D           EXE$GL_DYNAMIC_FLAGS          ********    X   02
CHP$_ACCLASS              = 00000005    D           EXE$GL_SYSUIC                 ********    X   02
CHP$_ACL                  = 0000000E    D           EXE$PROBER                    ********    X   02
CHP$_ACMODE               = 00000004    D           EXE$PROBEW                    ********    X   02
CHP$_ADDRIGHTS            = 00000007    D           EXE$SEARCH_RIGHT              0000054A RG D   02
CHP$_ALARMNAME            = 00000010    D           EXE$V_CLASS_PROT              ********    X   02
CHP$_AUDITNAME            = 0000000F    D           FINISH_ITEMS                  000000F0 R  D   02
CHP$_END                  = 00000[illegible]    D           GET_ITEM                      000000FE R  D   02
```

```
INDEX_TABLE               00000013 R  D  02    PR$_IPL             = 00000012    D
IPL$_ASTDEL             = 00000002    D        PRIVS_USED          = 000000E8    D
ITEM_ACCESS               000001BD R  D  02    PRV$V_BYPASS        = 0000001D    D
ITEM_ACCLASS              00000197 R  D  02    PRV$V_DOWNGRADE     = 00000021    D
ITEM_ACL                  000001F9 R  D  02    PRV$V_GRPPRV        = 000)0022    D
ITEM_ACMODE               000001C2 R  D  02    PRV$V_READALL       = 00000023    D
ITEM_ADDRIGHTS            0000021C R  D  02    PRV$V_SYSPRV        = 0000001C    D
ITEM_ALARMNAME            000001D9 R  D  02    PRV$V_UPGRADE       = 00000020    D
ITEM_AUDITNAME            000001D9 R  D  02    PSL$S_PRVMOD        = 00000002    D
ITEM_FLAGS                000001BD R  D  02    PSL$V_PRVMOD        = 00000016    D
ITEM_MATCHEDACE           000001D9 R  D  02    RETURN_ACCVIO         000000E4 R  D  02
ITEM_MAXCLASS             0000018E R  D  02    RETURN_ACCVIO1        00000182 R  D  02
ITEM_MINCLASS             00000188 R  D  02    RETURN_P1_BLOCK       0000043C R  D  02
ITEM_MODE                 000001F2 R  D  02    RETURN_STATUS         0000040D R  D  02
ITEM_MODES                000001AF R  D  02    RIGHTS_DESC         = 0000018C    D
ITEM_OWNER                000001BD R  D  02    RIGHTS_LIST         = 00000030    D
ITEM_PRIV                 000001AF R  D  02    SCH$GL_CURPCB         ********   X   02
ITEM_PRIVUSED             000001C7 R  D  02    SCH$LOCKR             ********   X   02
ITEM_PROT                 00000247 R  D  02    SCH$UNLOCK            ********   X   02
ITEM_RIGHTS               0000021C R  D  02    SS$_ACCVIO          = 0000000C    D
LOCAL_ARB               = 00000010    D        SS$_ACLFULL         = 000009F8    D
LOCAL_CHPCTL            = 000000B4    D        SS$_BADPARAM        = 00000014    D
LOCAL_CHPRET            = 000000C0    D        SS$_IVACL           = 000021E4    D
LOCAL_LENGTH            = 000001E4    D        SS$_NOENTRY         = 000009D8    D
LOCAL_ORB               = 0000005C    D        SS$_NOPRIV          = 00000024    D
MAX_ACL_DESC            = 00000014    D        SS$_NORMAL          = 00000001    D
MAX_CHP_CODE            = 00000012    D        SS$_NOSUCHID        = 000021EC    D
MAX_RIGHT_DESC          = 0000000B    D        SS$_RIGHTSFULL      = 000009E8    D
MIN_SIZE_TABLE            00000000 R  D  02    STRUCT_ADDR         = 00000000    D
NEXT_ITEM                 000001AC R  D  02    TMP_PC              = 00000072 R  D  02
NEXT_ITEM1                0000023C R  D  02    UIC$K_ID_FORMAT     = 00000002    D
OFFSET_TABLE              00000026 R  D  02    UIC$K_MATCH_ALL     = FFFFFFFF    D
ORB$B_FLAGS             = 0000000B    D        UIC$K_UIC_FORMAT    = 00000000    D
ORB$B_MODE              = 00000010    D        UIC$K_WILD_GROUP    = 00003FFF    D
ORB$C_LENGTH            = 00000058    D        UIC$K_WILD_MEMBER   = 0000FFFF    D
ORB$L_ACLFL             = 00000028    D        UIC$S_GROUP         = 0000000E    D
ORB$L_ACL_COUNT         = 00000028    D        UIC$V_FORMAT        = 0000001E    D
ORB$L_ACL_DESC          = 0000002C    D        UIC$V_GROUP         = 00000010    D
ORB$L_ACL_MUTEX         = 00000004    D
ORB$L_GRP_PROT          = 00000020    D
ORB$L_OWNER             = 00000000    D
ORB$L_OWN_PROT          = 0000001C    D
ORB$L_SYS_PROT          = 00000018    D
ORB$L_WOR_PROT          = 00000024    D
ORB$M_MODE_VECTOR       = 00000004    D
ORB$Q_MODE_PROT         = 00000010    D
ORB$R_MAX_CLASS         = 00000044    D
ORB$R_MIN_CLASS         = 00000030    D
ORB$S_MAX_CLASS         = 00000014    D
ORB$S_MIN_CLASS         = 00000014    D
ORB$V_ACL_QUEUE         = 00000001    D
ORB$V_CLASS_PROT        = 00000004    D
ORB$V_MODE_VECTOR       = 00000002    D
ORB$V_PROT_16           = 00000000    D
ORB$W_PROT              = 00000018    D
ORB_INDEX               = 00000001    D
PCB$L_ARB               = 0000008C    D
```

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

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)  01 (  1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD    WRT NOVEC BYTE
Y$EXEPAGED                      000007A7  (  1959.)  02 (  2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD    WRT NOVEC BYTE

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

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                        29    00:00:00.06    00:00:01.56
Command processing                   110    00:00:00.49    00:00:04.63
Pass 1                               411    00:00:15.29    00:00:47.90
Symbol table sort                      0    00:00:01.97    00:00:06.18
Pass 2                               350    00:00:05.47    00:00:19.56
Symbol table output                   25    00:00:00.16    00:00:00.42
Psect synopsis output                  2    00:00:00.03    00:00:00.03
Cross-reference output                 0    00:00:00.00    00:00:00.00
Assembler run totals                 929    00:00:23.47    00:01:20.28

The working set limit was 1950 pages.
91137 bytes (179 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1228 non-local and 94 local symbols.
1668 source lines were read in Pass 1, producing 86 object records in Pass 2.
33 pages of virtual memory were used to define 32 macros.

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

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                          13
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       15
TOTALS (all libraries)                                  28

1325 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

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

0382 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSCANCEL
LIS

SYSCANEVT
LIS

SYSCHGMOD
LIS

SYSCHKPRO
LIS

SYSCOMMON
LIS

SYSCREDEL
LIS

SYSCREPRC
LIS