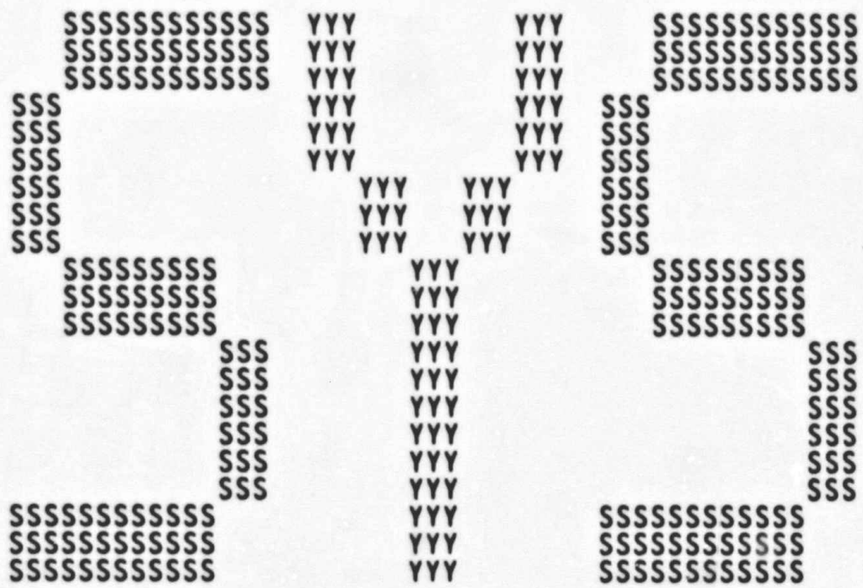
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

```
EEEEEEEEEE  XX      XX   SSSSSSSS  UU      UU  BBBBBBBB    RRRRRRRR     OOOOOO    UU      UU  TTTTTTTTTT
EEEEEEEEEE  XX      XX   SSSSSSSS  UU      UU  BBBBBBBB    RRRRRRRR     OOOOOO    UU      UU  TTTTTTTTTT
EE          XX      XX  SS         UU      UU  BB      BB  RR      RR  OO      OO  UU      UU      TT
EE          XX      XX  SS         UU      UU  BB      BB  RR      RR  OO      OO  UU      UU      TT
EE            XX  XX    SS         UU      UU  BB      BB  RR      RR  OO      OO  UU      UU      TT
EE            XX  XX    SS         UU      UU  BB      BB  RR      RR  OO      OO  UU      UU      TT
EEEEEEEE        XX        SSSSSS   UU      UU  BBBBBBBB    RRRRRRRR    OO      OO  UU      UU      TT
EEEEEEEE        XX        SSSSSS   UU      UU  BBBBBBBB    RRRRRRRR    OO      OO  UU      UU      TT
EE            XX  XX            SS UU      UU  BB      BB  RR  RR      OO      OO  UU      UU      TT
EE            XX  XX            SS UU      UU  BB      BB  RR  RR      OO      OO  UU      UU      TT
EE          XX      XX          SS UU      UU  BB      BB  RR    RR    OO      OO  UU      UU      TT      ....
EE          XX      XX          SS UU      UU  BB      BB  RR    RR    OO      OO  UU      UU      TT      ....
EEEEEEEEEE  XX      XX  SSSSSSSS   UUUUUUUUUU  BBBBBBBB    RR      RR   OOOOOO    UUUUUUUUUU      TT      ....
EEEEEEEEEE  XX      XX  SSSSSSSS   UUUUUUUUUU  BBBBBBBB    RR      RR   OOOOOO    UUUUUUUUUU      TT      ....

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

Table of contents

(2)	82	Declarations
(3)	111	Local Macro Definitions and storage
(6)	207	CHECK PROCESS RESOURCE REQUEST
(7)	302	CHECK ACCESS PROTECTION
(8)	409	CLEANUP AN OBJECT RIGHTS BLOCK
(9)	469	INSERT ENTRY IN TIME DEPENDENT SCHEDULER QUEUE
(10)	508	MAXIMIZE ACCESS MODE
(11)	535	REMOVE ENTRY FROM TIME DEPENDENT SCHEDULER QUEUE
(11)	603	EXE$PROBER - Check read accessibility of user buffer
(11)	671	EXE$PROBEW - Check write accessibility of user buffer
(11)	733	EXE$PROBER_DSC, EXE$PROBEW_DSC - Check buffer accessibility
(12)	818	EXE$VAL_IDNAME - Validate ID name

```
0000     1          .TITLE  EXSUBROUT - EXECUTIVE SUPPORT SUBROUTINES
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;******************************************************************************
0000     6 ;*                                                                            *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                   *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
0000     9 ;*  ALL RIGHTS RESERVED.                                                      *
0000    10 ;*                                                                            *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16 ;*  TRANSFERRED.                                                              *
0000    17 ;*                                                                            *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20 ;*  CORPORATION.                                                              *
0000    21 ;*                                                                            *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
0000    24 ;*                                                                            *
0000    25 ;*                                                                            *
0000    26 ;******************************************************************************
0000    27 ;
0000    28 ; Abstract:
0000    29 ;
0000    30 ;       EXECUTIVE SUPPORT SUBROUTINES
0000    31 ;
0000    32 ; Author:
0000    33 ;
0000    34 ;       D. N. CUTLER 17-AUG-76
0000    35 ;
0000    36 ; MODIFIED BY:
0000    37 ;
0000    38 ;       V03-012 TMK0001         Todd M. Katz            11-Apr-1984
0000    39 ;               Add the routine EXE$CLEANUP_ORB. This routine cleans up all
0000    40 ;               structures referenced by an ORB. Also, change all occurrances
0000    41 ;               of BSBWs to EXE$CHKPRO_ALT in the EXE$CHKxxxACCES routines to
0000    42 ;               EXE$CHKPRO_INT since LMP0221 missed a couple of them.
0000    43 ;
0000    44 ;       V03-011 LMP0221         L. Mark Pilant,         7-Apr-1984  11:10
0000    45 ;               Move EXE$CHKPRO_ALT to SYSCHKPRO.MAR, also change the name
0000    46 ;               to EXE$CHKPRO_INT as it is now the real thing.  Modify the
0000    47 ;               EXE$CHKxxxACCES routine to use the ORB from the UCB.
0000    48 ;
0000    49 ;       V03-010 LMP0214         L. Mark Pilant,         24-Mar-1984  10:54
0000    50 ;               Add support for the new EXE$CHKPRO_INT interface.  This
0000    51 ;               includes adding the temporary routine EXE$CHKPRO_ALT so
0000    52 ;               as not to totally eliminate the old interface.
0000    53 ;
0000    54 ;       V03-009 ACG0408         Andrew C. Goldstein,    20-Mar-1984  17:08
0000    55 ;               Remove obsolete EXE$GETACCESS
0000    56 ;
0000    57 ;       V03-008 LMP0190         L. Mark Pilant,         6-Feb-1984  9:07
```

```
0000     58 ;                Remove the assumption that a CHIP block has 16 byte granularity.
0000     59 ;
0000     60 ;        V03-007 LMP0185         L. Mark Pilant,         19-Jan-1984  15:25
0000     61 ;                Modify the interface to the EXE$CHKxxxACCES routines to allow
0000     62 ;                additional protection information to be checked.
0000     63 ;
0000     64 ;        V03-006 RSH0083         R. Scott Hanna          22-Nov-1983
0000     65 ;                Modify return status in EXE$VAL_IDNAME from SS$_NOSUCHID
0000     66 ;                to SS$_IVIDENT.
0000     67 ;
0000     68 ;        V03-005 RSH0044         R. Scott Hanna          12-Jul-1983
0000     69 ;                Complete EXE$VAL_IDNAME
0000     70 ;
0000     71 ;        V03-004 RSH0042         R. Scott Hanna          22-Jun-1983
0000     72 ;                Add EXE$VAL_IDNAME
0000     73 ;
0000     74 ;        V03-003 WMC0001         Wayne Cardoza           18-Apr-1983
0000     75 ;                Add EXE$CHKEXEACCES.
0000     76 ;
0000     77 ;        V03-002 KDM0002         Kathleen D. Morse       28-Jun-1982
0000     78 ;                Add $PRVDEF and $VADEF.
0000     79 ;
0000     80 ;--
```

```
0000     82          .SBTTL  Declarations
0000     83
0000     84 ;
0000     85 ; MACRO LIBRARY CALLS
0000     86 ;
0000     87
0000     88          $ACBDEF                 ;DEFINE ACB OFFSETS
0000     89          $ACEDEF                 ;DEFINE ACE TYPES & OFFSETS
0000     90          $ACLDEF                 ;DEFINE ACCESS CONTROL LIST BLOCK
0000     91          $ARBDEF                 ;DEFINE ACCESS RIGHTS BLOCK
0000     92          $ARMDEF                 ;DEFINE ACCESS RIGHTS MASK
0000     93          $CHPDEF                 ;DEFINE PROTECTION CHECK CODES
0000     94          $CHPCTLDEF              ;$CHKPRO CONTROL BLOCK OFFSETS
0000     95          $CHPRETDEF              ;$CHKPRO RETURN ARG BLOCK OFFSETS
0000     96          $DSCDEF                 ;DEFINE CHARACTER DESCRIPTOR BLOCK
0000     97          $IPLDEF                 ;DEFINE PRIORITY LEVELS
0000     98          $JIBDEF                 ;DEFINE JOB INFORMATION BLOCK
0000     99          $KGBDEF                 ;DEFINE KEY GRANT BLOCK
0000    100          $ORBDEF                 ;OBJECTS RIGHTS BLOCK OFFSETS
0000    101          $PCBDEF                 ;DEFINE PCB OFFSETS
0000    102          $PRDEF                  ;DEFINE PROCESSOR REGISTERS
0000    103          $PRVDEF                 ;DEFINE PRIVILEGES
0000    104          $PSLDEF                 ;DEFINE PROCESSOR STATUS FIELDS
0000    105          $RSNDEF                 ;DEFINE RESOURCE WAIT NUMBERS
0000    106          $SSDEF                  ;DEFINE SYSTEM STATUS VALUES
0000    107          $TQEDEF                 ;DEFINE TQE OFFSETS
0000    108          $UCBDEF                 ;DEFINE UNIT CONTROL BLOCK
0000    109          $VADEF                  ;DEFINE VIRTUAL ADDRESS FIELDS
```

```
                        .SBTTL  Local Macro Definitions and storage
                ;
                ;       This macro is used to define masks for the scan table. It defines
                ;       local symbols for the mask bit position, mask longword, and symbols
                ;       used to build the scan table with the TABLE_BLD macro. Each invocation
                ;       of the macro allocates successive bits to be used in the table. The
                ;       macro is invoked as follows:
                ;
                ;               MASK_DEF name,<<range>,...>
                ;
                ;               Where range is a single decimal number or two numbers
                ;               separated by a comma. These numbers specify a single or
                ;               contiguous range of byte offsets in the scan table for
                ;               which the scan bit is to be set.
                ;
                ;       Example:
                ;
                ;               MASK_DEF ID_NAME,<<65,90>,<97,122>,<48,57>,<36>,<95>>
                ;
                ;               This would define the symbols MASK_V_ID_NAME and
                ;               MASK_M_ID_NAME. It will also cause the bit MASK_V_ID_NAME
                ;               to be set in the scan table at offsets which correspond to
                ;               the characters A-Z, a-z, $, and _.
                ;
0000    135     .MACRO  MASK_DEF MASK_NAME,LIST
0000    136
0000    137     .MACRO  MASK_DEF_2 RANGE_LOW,RANGE_HIGH
0000    138     .IF     B       <RANGE_HIGH>
0000    139     $$ = 1
0000    140     .IFF
0000    141     $$ = RANGE_HIGH-RANGE_LOW+1
0000    142     .ENDC
0000    143     CHAR = RANGE_LOW
0000    144     .REPEAT $$
0000    145     MASK_DEF_3 \CHAR
0000    146     CHAR = CHAR + 1
0000    147     .ENDR
0000    148     .ENDM
0000    149
0000    150     .MACRO  MASK_DEF_3 $$$
0000    151     .IF     NDF     CHAR'$$$
0000    152     CHAR'$$$ = 1@MASK_K_NUM
0000    153     .IFF
0000    154     CHAR'$$$ = CHAR'$$$ ! <1@MASK_K_NUM>
0000    155     .ENDC
0000    156     .ENDM
0000    157
0000    158     .IF     NDF     MASK_K_NUM
0000    159     MASK_K_NUM = 0
0000    160     .ENDC
0000    161     MASK_V_'MASK_NAME = MASK_K_NUM
0000    162     MASK_M_'MASK_NAME = 1@MASK_K_NUM
0000    163     .IRP    CLASS,<LIST>
0000    164     MASK_DEF_2 CLASS
0000    165     .ENDR
0000    166     MASK_K_NUM = MASK_K_NUM + 1
0000    167     .ENDM
```

```
                        ;
                        ;       This macro uses symbols defined by the MASK_DEF macro to build
                        ;       the scan table.
                        ;

                        .MACRO  TABLE_BLD

                        .MACRO  TABLE_BLD_2 $$
                        .IF     NDF     CHAR'$$
                        .BYTE   0
                        .IFF
                        .BYTE   CHAR'$$
                        .ENDC
                        .ENDM

                        CHAR = 0
                        .REPEAT 256
                        TABLE_BLD_2 \CHAR
                        CHAR = CHAR + 1
                        .ENDR
                        .ENDM

                        ;
                        ;       Define the scan table mask bits
                        ;
                        MASK_DEF        NUMBER,<<48,57>>                                ;0-9
                        MASK_DEF        ID_NAME,<<65,90>,<97,122>,<48,57>,<36>,<95>>    ;A-Z, a-z, 0

                        ;
                        ;       Build the scan table
                        ;

                        .PSECT  Y$EXEPAGED

SCAN_TABLE:
                        TABLE_BLD
```

```
                     0100   207          .SBTTL  CHECK PROCESS RESOURCE REQUEST
                     0100   208
                     0100   209          .ENABL  LSB
                     0100   210 ;+
                     0100   211 ; EXE$BUFFRQUOTA - CHECK BUFFER BYTE COUNT QUOTA AND CONDITIONALLY WAIT
                     0100   212 ; EXE$BUFQUOPRC - CHECK PROCESS BUFFER QUOTA ONLY AND CONDITIONALLY WAIT
                     0100   213 ; EXE$MULTIQUOTA - CHECK MULTI-UNIT RESOURCE REQUEST AND CONDITIONALLY WAIT
                     0100   214 ; EXE$SNGLEQUOTA - CHECK SINGLE-UNIT RESOURCE REQUEST AND CONDITIONALLY WAIT
                     0100   215 ;
                     0100   216 ; THESE ROUTINES ARE CALLED TO CHECK IF A PROCESS HAS SUFFICIENT RESIDUAL
                     0100   217 ; QUOTA TO GRANT A RESOURCE. IF SUFFICIENT QUOTA IS NOT AVAILABLE AND THE
                     0100   218 ; PROCESS IS ENABLED FOR IMPLIED RESOURCE WAIT, THEN THE PROCESS IS ENTERED
                     0100   219 ; IN A WAIT STATE TO WAIT FOR SUFFICIENT QUOTA TO BECOME AVAILABLE.
                     0100   220 ;
                     0100   221 ; INPUTS:
                     0100   222 ;
                     0100   223 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0100   224 ;
                     0100   225 ; IF ENTRY AT EXE$BUFFRQUOTA, EXE$BUFQUOPRC, OR EXE$MULTIQUOTA, THEN
                     0100   226 ;
                     0100   227 ;       R1 = NUMBER OF UNITS OF RESOURCE REQUESTED.
                     0100   228 ;
                     0100   229 ; IF ENTRY AT EXE$MULTIQUOTA OR EXE$SNGLEQUOTA, THEN
                     0100   230 ;
                     0100   231 ;       R2 = ADDRESS OF QUOTA WORD CONTAINING REMAINING UNITS FOR SPECIFIED
                     0100   232 ;            RESOURCE.
                     0100   233 ;
                     0100   234 ;       NOTICE: EXE$MULTIQUOTA and EXE$SNGLEQUOTA cannot be used to check
                     0100   235 ;               Buffered I/O Byte Count Quota, since these quota and count
                     0100   236 ;               values  are stored in longwords.  EXE$BUFFRQUOTA or
                     0100   237 ;               EXE$BUFQUOPRC must be use to check Buffered I/O Byte Count
                     0100   238 ;               Quota.
                     0100   239 ;
                     0100   240 ; OUTPUTS:
                     0100   241 ;
                     0100   242 ;       R0 LOW BIT CLEAR INDICATES CHECK FAILURE WITH CALLING IPL PRESERVED.
                     0100   243 ;
                     0100   244 ;               R0 = SS$_EXQUOTA = QUOTA EXCEEDED OR REQUEST GREATER THAN
                     0100   245 ;                    MAXIMUM PROCESS OR SYSTEM QUOTA.
                     0100   246 ;
                     0100   247 ;       R0 LOW BIT SET INDICATES SUCCESS WITH IPL SET TO AST DELIVERY
                     0100   248 ;            LEVEL.
                     0100   249 ;
                     0100   250 ;               R0 = SS$_NORMAL = NORMAL COMPLETION.
                     0100   251 ;
                     0100   252 ;       The following table shows how the various routines treat R1 - R3.
                     0100   253 ;
                     0100   254 ;       Routine          R1      R2      R3
                     0100   255 ;       EXE$BUFFRQUOTA    P       D       D
                     0100   256 ;       EXE$BUFQUOPRC     P       D       D       P ==> Preserved
                     0100   257 ;       EXE$MULTIQUOTA    P       P       D       D ==> Destroyed
                     0100   258 ;       EXE$SNGLEQUOTA    D       P       D
                     0100   259 ;-
                     0100   260
                00000000    261          .PSECT  AEXENONPAGED
                    0000    262 EXE$BUFFRQUOTA::                            ;CHECK BUFFER BYTE COUNT QUOTA
50   00000000'EF    3C  0000 263          MOVZWL  IOC$GW_MAXBUF,R0          ; Get max system buffer size.
```

```
                                                H 11
                     50   51   D1  0007   264          CMPL    R1,R0                   ;REQUEST GREATER THAN SYSTEM MAXIMUM?
                          12   1A  000A   265          BGTRU   10$                     ;IF GTRU YES
                               000C   266 EXE$BUFQUOPRC::                             ;CHECK PROCESS BUFFER QUOTA ONLY
                50   0080 C4   D0  000C   267          MOVL    PCB$L_JIB(R4),R0        ;GET ADDRESS OF JOB INFORMATION BLOCK
                  52   20 A0   9E  0011   268          MOVAB   JIB$L_BYTCNT(R0),R2     ;SET ADDRESS OF BYTE COUNT QUOTA
                     53   20   D0  0015   269          MOVL    #32, R3                 ;ESTABLISH LONGWORD QUOTA CHECKING.
                   24 A0   51   D1  0018   270          CMPL    R1,JIB$L_BYTLM(R0)      ;REQUEST GREATER THAN PROCESS MAXIMUM?
                          0A   1B  001C   271          BLEQU   15$                     ;IF LEQU NO
                     50   1C   3C  001E   272 10$:      MOVZWL  #SS$_EXQUOTA,R0         ;SET QUOTA EXCEEDED
                          05   0021   273          RSB                             ;
                               0022   274 EXE$SNGLEQUOTA::                          ;CHECK SINGLE-UNIT RESOURCE REQUEST
                     51   01   D0  0022   275          MOVL    #1,R1                   ;SET NUMBER OF UNITS OF RESOURCE REQUESTED
                               0025   276 EXE$MULTIQUOTA::                          ;CHECK MULTI-UNIT RESOURCE REQUEST
                     53   10   D0  0025   277          MOVL    #16, R3                 ;Establish word length quota checking.
                          7E   DC  0028   278 15$:      MOVPSL  -(SP)                   ;Save PSL for possible resource wait.
                               002A   279          DSBINT  #IPL$_SYNCH             ;Synchronize system data base access,
                               0030   280                                          ;and save current IPL on stack.
           51   62   53   00   ED  0030   281          CMPZV   #0, R3, (R2), R1        ;Compare either word or longword
                               0035   282                                          ;using number of bits value in R3.
                          0E   1F  0035   283          BLSSU   30$                     ;Branch if insufficent resourse left.
                     50   01   3C  0037   284          MOVZWL  #SS$_NORMAL,R0          ;Enough quota left; normal completion.
                     6E   02   D0  003A   285          MOVL    #IPL$_ASTDEL,(SP)       ;Insure exit IPL is 2, so that checked
                               003D   286                                          ;resource is protected from AST's.
                               003D   287                                          ;SS$_NORMAL & SS$_EXQUOTA common exit
                               003D   288 20$:      ENBINT                          ;Restore IPL as appropriate.
                5E   04 AE   9E  0040   289          MOVAB   4(SP),SP                ;Remove saved PSL from stack.
                          05   0044   290          RSB                             ;Return to caller.
                               0045   291
                     50   1C   3C  0045   292 30$:      MOVZWL  #SS$_EXQUOTA, R0        ;Setup possible exceeded quota return
           F0 24 A4   0A   E0  0048   293          BBS     #PCB$V_SSRWAIT, -       ;code, and branch if resource wait
                               004D   294                  PCB$L_STS(R4), 20$      ;mode is disabled.  (NB: restored IPL
                               004D   295                                          ;will be that of our caller.)
                50   01   3C  004D   296          MOVZWL  #RSN$_ASTWAIT, R0       ;Set AST wait resource number.
                5E   04 AE   9E  0050   297          MOVAB   4(SP), SP               ;Strip saved caller's IPL from stack.
                     FFA9'   30  0054   298          BSBW    SCH$RWAIT               ;Wait; adding PC stacked caller's PSL.
                          CF   11  0057   299          BRB     15$                     ;Then try again.
                               0059   300          .DSABL  LSB
```

```
                                                I 11
0059   302          .SBTTL CHECK ACCESS PROTECTION
0059   303 ;+
0059   304 ; EXE$CHKCREACCES - CHECK CREATE ACCESS
0059   305 ; EXE$CHKDELACCES - CHECK DELETE ACCESS
0059   306 ; EXE$CHKLOGACCES - CHECK LOGICAL I/O FUNCTION ACCESS
0059   307 ; EXE$CHKPHYACCES - CHECK PHYSICAL I/O FUNCTION ACCESS
0059   308 ; EXE$CHKRDACCES - CHECK READ ACCESS
0059   309 ; EXE$CHKWRTACCES - CHECK WRITE ACCESS
0059   310 ; EXE$CHKEXEACCES - CHECK EXECUTE ACCESS (IMPLIED BY READ ACCESS)
0059   311 ;
0059   312 ; THESE ROUTINES RETURN A TRUE OR FALSE VALUE ON THE PROTECTION
0059   313 ; INFORMATION SUPPLIED.
0059   314 ;
0059   315 ; INPUTS:
0059   316 ;
0059   317 ;       R0 = ADDRESS OF THE AGENT'S RIGHTS BLOCK
0059   318 ;       R1 = ADDRESS OF THE OBJECT'S RIGHTS BLOCK
0059   319 ;       R5 = 0
0059   320 ;
0059   321 ;            OR
0059   322 ;
0059   323 ;       R4 = ADDRESS OF THE ACCESSOR'S PCB
0059   324 ;       R5 = ADDRESS OF THE OBJECT'S UCB
0059   325 ;
0059   326 ; OUTPUTS:
0059   327 ;
0059   328 ;       R0 = SS$_NORMAL FOR ACCESS ALLOWED
0059   329 ;       R0 = SS$_NOPRIV FOR ACCESS DENIED
0059   330 ;
0059   331 ;       R2, R3, AND R4 ARE PRESERVED ACROSS CALL.
0059   332 ;-
0059   333
0059   334 ; FOLLOWING ARE THE VERIOUS OFFSETS FOR LOCAL STORAGE.
0059   335
00000100   336          .PSECT  Y$EXEPAGED
0100   337
0100   338          .ENABL  LSB
0100   339 EXE$CHKDELACCES::                           ;CHECK DELETE ACCESS
0100   340 EXE$CHKLOGACCES::                           ;CHECK LOGICAL I/O FUNCTION ACCESS
7E 08 3C 0100   341          MOVZWL  #ARM$M_DELETE,-(SP)     ;SET NEEDED ACCESS
   12 11 0103   342          BRB     10$                     ;
0105   343 EXE$CHKCREACCES::                           ;CHECK CREATE ACCESS
0105   344 EXE$CHKPHYACCES::                           ;CHECK PHYSICAL I/O FUNCTION ACCESS
7E 04 3C 0105   345          MOVZWL  #ARM$M_EXECUTE,-(SP)    ;SET NEEDED ACCESS
   0D 11 0108   346          BRB     10$                     ;
010A   347 EXE$CHKRDACCES::                           ;CHECK READ ACCESS
7E 01 3C 010A   348          MOVZWL  #ARM$M_READ,-(SP)       ;SET NEEDED ACCESS
   08 11 010D   349          BRB     10$                     ;
010F   350 EXE$CHKEXEACCES::                          ;CHECK EXECUTE (IMPLIED BY READ) ACCESS
7E 05 3C 010F   351          MOVZWL  #ARM$M_READ!ARM$M_EXECUTE,-(SP) ;SET NEEDED ACCESS
   03 11 0112   352          BRB     10$                     ;
0114   353 EXE$CHKWRTACCES::                          ;CHECK WRITE ACCESS
7E 02 3C 0114   354          MOVZWL  #ARM$M_WRITE,-(SP)      ;SET NEEDED ACCESS
0117   355
0117   356 ; NOW FOR THE FUN.  IN THE NORMAL CASE, THE INFORMATION IS SIMPLY PASSED
0117   357 ; THROUGH TO EXE$CHKPRO_INT.  HOWEVER, TO SUPPORT THE OLD INTERFACE, IT
0117   358 ; WILL BE NECESSARY BUILD THE CHKPRO CONTROL BLOCK AND DETERMINE THE ARB
```

```
                                J 11
                     0117   359 ; AND ORB ADDRESSES.  IN EITHER CASE, EXE$CHKPRI_INT RETURNS SS$_NORMAL IF
                     0117   360 ; ACCESS IS ALLOWED OT SS$_NOPRIV IF ACCESS IS DENIED.
                     0117   361
           1E   BB   0117   362 10$:    PUSHR   #^M<R1,R2,R3,R4>        ;PRESERVE NEEDED REGISTERS
                     0119   363
                     0119   364 ; IF THIS IS THE OLD INTERFACE, DETERMINE THE ARB AND ORB ADDRESSES.  THEN
                     0119   365 ; FABRICATE THE CHPCTL BLOCK ON THE STACK.  IF THIS IS THE NEW INTERFACE,
                     0119   366 ; SIMPLY PASS THE REGISTERS ON THROUGH.  IN EITHER CASE, THE CONTROL BLOCK
                     0119   367 ; MUST BE BUILT NOW.
                     0119   368
                     0119   369         ASSUME  CHPCTL$C_LENGTH EQ 12
                     0119   370
           7E   7C   0119   371         CLRQ    -(SP)                   ;BUILD CONTROL BLOCK ON THE STACK
           7E   D4   011B   372         CLRL    -(SP)
        52 5E   D0   011D   373         MOVL    SP,R2                   ;SAVE THE ADDRESS FOR LATER
           53   D4   0120   374         CLRL    R3                      ;NO RETURN ARG BLOCK
                     0122   375
                     0122   376         ASSUME  CHP$M_READ EQ CHPCTL$M_READ
                     0122   377         ASSUME  CHP$M_USEREADALL EQ CHPCTL$M_USEREADALL
                     0122   378         ASSUME  CHP$M_WRITE EQ CHPCTL$M_WRITE
                     0122   379
     62   1C AE   D0   0122   380         MOVL    28(SP),CHPCTL$L_ACCESS(R2)      ;SET THE ACCESS DESIRED
  04 A2      05   D0   0126   381         MOVL    #CHP$M_READ!CHP$M_USEREADALL,CHPCTL$L_FLAGS(R2) ;SET FOR READ
     62    FA 8F   93   012A   382         BITB    #^C<ARM$M_READ!ARM$M_EXECUTE>,CHPCTL$L_ACCESS(R2)
           04   13   012E   383         BEQL    20$                     ;XFER IF ONLY READING
  04 A2    06   8C   0130   384         XORB    #CHP$M_WRITE!CHP$M_USEREADALL,CHPCTL$L_FLAGS(R2)        ;ELSE SET WR
                     0134   385                                         ;AND NO READALL
                     0134   386
           55   D5   0134   387 20$:    TSTL    R5                      ;ANY UCB GIVEN?
           0B   13   0136   388         BEQL    30$                     ;XFER IF NOT, REGISTERS ALL SET
     50    008C C4   0138   389         MOVL    PCB$L_ARB(R4),R0        ;NOTE THE ARB ADDRESS
        51    1C A5   013D   390         MOVL    UCB$L_ORB(R5),R1        ;  AND THE ORB ADDRESS
           10   11   0141   391         BRB     40$                     ;GO DO THE PROTECTION CHECK
                     0143   392
                     0143   393 ; NOW DO THE PROTECTION CHECK.
                     0143   394
           50   D5   0143   395 30$:    TSTL    R0                      ;WAS AN ARB GIVEN?
           0C   12   0145   396         BNEQ    40$                     ;XFER IF SO
54   00000000'EF   D0   0147   397         MOVL    CTL$GL_PCB,R4           ;ELSE GET THE PROCESS DEFAULT
     50    008C C4   D0   014E   398         MOVL    PCB$L_ARB(R4),R0        ;  AGENT'S RIGHTS BLOCK
      00000000'EF   16   0153   399 40$:    JSB     EXE$CHKPRO_INT          ;DO THE PROTECTION CHECK
                     0159   400
                     0159   401 ; FINALLY, RETURN WITH THE STATUS POSTED IN R0.
                     0159   402
        5E   0C   C0   0159   403         ADDL2   #CHPCTL$C_LENGTH,SP     ;CLEAN OFF THE TEMP CONTROL BLOCK
           1E   BA   015C   404         POPR    #^M<R1,R2,R3,R4>        ;RESTORE SAVED RESISTERS
           8E   D5   015E   405         TSTL    (SP)+                   ;POP OFF ACCESS MASK
           05   0160   406         RSB                             ;RETURN TO CALLER
                     0161   407         .DSABL  LSB
```

```
                                0161   409          .SBTTL  CLEANUP AN OBJECT RIGHTS BLOCK
                                0161   410 ;+
                                0161   411 ; EXE$CLEANUP_ORB - DELETE ALL STRUCTURES REFERENCED BY AN OBJECT RIGHTS BLOCK
                                0161   412 ;
                                0161   413 ; THIS ROUTINE IS CALLED TO DELETE ALL STRUCTURES REFERENCED BY AN OBJECT RIGHTS
                                0161   414 ; BLOCK.  THIS ROUTINE DOES NOT DELETE THE ORB ITSELF.  THE ASSUMPTION IS MADE
                                0161   415 ; THAT ALL STRUCTURES TO BE DELETED RESIDE IN PAGED-POOL.  BEFORE EACH STRUCTURE
                                0161   416 ; IS DEALLOCATED, ACCESS TO IS IS SYNCHRONIZED BY WHATEVER METHOD IS REQUIRED.
                                0161   417 ; THE LIST OF ORB REFERENCED STRUCTURES WHICH THIS ROUTINE DISPOSES OF IS:
                                0161   418 ;
                                0161   419 ; 1. LIST OF ACLS ASSOCIATED WITH THIS OBJECT RIGHTS BLOCK.
                                0161   420 ;
                                0161   421 ; INPUTS:
                                0161   422 ;
                                0161   423 ;       R1 = ADDRESS OF OBJECT RIGHTS BLOCK.
                                0161   424 ;
                                0161   425 ; OUTPUTS:
                                0161   426 ;
                                0161   427 ;       R0-R3 ARE DESTROYED.
                                0161   428 ;       ALL OTHER REGISTERS ARE PRESERVED.
                                0161   429 ;       ALL STRUCTURES REFERENCED BY THE ORB ARE DEALLOCATED TO PAGED POOL.
                                0161   430 ;-
                                0161   431
                                0161   432 EXE$CLEANUP_ORB::                          ;DELETE ALL ORB REFERENCED STRUCTURES
                     7E   54    7D 0161   433          MOVQ    R4,-(SP)               ;SAVE REGISTERS
                                0164   434          SAVIPL                         ;SAVE IPL ON ENTRY
                     55   51    D0 0167   435          MOVL    R1,R5                  ;MOVE ORB ADDRESS INTO R5
                54   00000000'EF  D0 016A   436          MOVL    CTL$GL_PCB,R4          ;RETRIEVE PCB ADDRESS
                                0171   437
                                0171   438 ;
                                0171   439 ; DELETE THE ACLS WHICH ARE ASSOCIATED WITH THIS OBJECT RIGHTS BLOCK.  ACCESS
                                0171   440 ; TO THE ACLS IS CONTROLLED BY A MUTEX RESIDING WITHIN THE ORB ITSELF.  THIS
                                0171   441 ; MUTEX IS LOCKED FOR WRITING BEFORE REFERENCING THE ACL LISTHEAD.  IF THE ACLS
                                0171   442 ; ARE NOT IN A QUEUE FORMAT BUT ARE IN A DESCRIPTOR STYLE FORMAT, THEN THEIR
                                0171   443 ; DELETION IS BYPASSED.
                                0171   444 ;
                                0171   445
                       01   E1  0171   446          BBC     #ORB$V_ACL_QUEUE,-     ;SKIP ACL QUEUE DELETION IF THE ACLS
                  1D 0B A5      0173   447                  ORB$B_FLAGS(R5),30$    ;ARE IN A DESCRIPTOR STYLE FORMAT
                                0176   448
                50   04 A5   9E 0176   449          MOVAB   ORB$L_ACL_MUTEX(R5),R0 ;RETRIEVE ADDRESS OF ACL MUTEX
                     FE83'   30 017A   450          BSBW    SCH$LOCKW              ;LOCK ACL MUTEX FOR WRITING
                                017D   451
                50   28 B5   0F 017D   452 10$:     REMQUE  @ORB$L_ACLFL(R5),R0    ;REMOVE NEXT ACL FROM QUEUE
                       09    1D 0181   453          BVS     20$                    ;DONE IF QUEUE IS EMPTY
                51   08 A0   3C 0183   454          MOVZWL  ACL$W_SIZE(R0),R1      ;RETRIEVE SIZE OF ACL
                     FE76'   30 0187   455          BSBW    EXE$DEAPGDSIZ          ;DELETE THE ACL
                       F1    11 018A   456          BRB     10$                    ;CONTINUE UNTIL ACL QUEUE IS EMPTY
                                018C   457
                50   04 A5   9E 018C   458 20$:     MOVAB   ORB$L_ACL_MUTEX(R5),R0 ;RETRIEVE ADDRESS OF ACL MUTEX
                     FE6D'   30 0190   459          BSBW    SCH$UNLOCK             ;UNLOCK ACL MUTEX
                                0193   460
                                0193   461 ;
                                0193   462 ; RESTORE THE ENVIRONMENT TO WHAT IT WAS ON ROUTINE ENTRY AND EXIT.
                                0193   463 ;
                                0193   464
                                0193   465 30$:     ENBINT                         ;RESTORE IPL
```

```
                54   8E   7D  0196   466          MOVQ    (SP)+,R4                ;RESTORE REGISTERS
                          05  0199   467          RSB                             ;RETURN
```

```
                     019A   469          .SBTTL  INSERT ENTRY IN TIME DEPENDENT SCHEDULER QUEUE
                     019A   470 ;+
                     019A   471 ; EXE$INSTIMQ - INSERT ENTRY IN TIME DEPENDENT SCHEDULER QUEUE
                     019A   472 ;
                     019A   473 ; THIS ROUTINE IS CALLED TO INSERT AN ENTRY IN THE TIME DEPENDENT SCHEDULER
                     019A   474 ; QUEUE. THE ENTRY IS THREADED INTO THE QUEUE ACCORDING TO ITS DUE TIME.
                     019A   475 ; THE QUEUE IS ORDERED SUCH THAT THE MOST IMMINENT ENTRIES ARE AT THE FRONT
                     019A   476 ; OF THE QUEUE.
                     019A   477 ;
                     019A   478 ; INPUTS:
                     019A   479 ;
                     019A   480 ;       R0 = LOW ORDER PART OF EXPIRATION TIME.
                     019A   481 ;       R1 = HIGH ORDER PART OF EXPIRATION TIME.
                     019A   482 ;       R5 = ADDRESS OF ENTRY TO INSERT IN TIME QUEUE.
                     019A   483 ;
                     019A   484 ;       IPL MUST BE IPL$_TIMER.
                     019A   485 ;
                     019A   486 ; OUTPUTS:
                     019A   487 ;
                     019A   488 ;       SPECIFIED ENTRY IS INSERTED INTO THE TIME DEPENDENT SCHEDULER QUEUE
                     019A   489 ;       ACCORDING TO ITS DUE TIME.
                     019A   490 ;-
                     019A   491
                     0000   492          .PSECT
                     0000   493 EXE$INSTIMQ::                            ;INSERT ENTRY IN TIME QUEUE
  18 A5   50   7D    0000   494          MOVQ    R0,TQE$Q_TIME(R5)       ;SET ABSOLUTE DUE TIME
53   0000'CF   DE    0004   495          MOVAL   W^EXE$GL_TQFL,R3        ;GET ADDRESS OF TIME QUEUE LISTHEAD
     52   53   D0    0009   496          MOVL    R3,R2                   ;COPY ADDRESS OF TIME QUEUE LISTHEAD
  52   04 A2   D0    000C   497 10$:      MOVL    TQE$L_TQBL(R2),R2       ;GET ADDRESS OF NEXT ENTRY
     52   53   D1    0010   498          CMPL    R3,R2                   ;END OF QUEUE?
          0E   13    0013   499          BEQL    20$                     ;IF EQL YES
  1C A2   51   D1    0015   500          CMPL    R1,TQE$Q_TIME+4(R2)     ;COMPARE HIGH ORDER PARTS OF TIME
          F1   1F    0019   501          BLSSU   10$                     ;IF LSSU NEW ENTRY MORE IMMINENT
          06   1A    001B   502          BGTRU   20$                     ;IF GTRU NEW ENTRY LESS IMMINENT
  18 A2   50   D1    001D   503          CMPL    R0,TQE$Q_TIME(R2)       ;COMPARE LOW ORDER PART OF TIME
          E9   1F    0021   504          BLSSU   10$                     ;IF LSSU NEW ENTRY MORE IMMINENT
     62   65   0E    0023   505 20$:      INSQUE  TQE$L_TQFL(R5),TQE$L_TQFL(R2) ;INSERT NEW ENTRY IN TIME QUEUE
               05    0026   506          RSB                             ;
```

```
                                0027   508          .SBTTL  MAXIMIZE ACCESS MODE
                                0027   509 ;+
                                0027   510 ; EXE$MAXACMODE - MAXIMIZE ACCESS MODE
                                0027   511 ;
                                0027   512 ; THIS ROUTINE IS CALLED TO MAXIMIZE A SPECIFIED ACCESS MODE WITH THE PREVIOUS
                                0027   513 ; MODE FIELD OF THE CURRENT PSL.
                                0027   514 ;
                                0027   515 ; INPUTS:
                                0027   516 ;
                                0027   517 ;       R0 = ACCESS MODE TO MAXIMIZE WITH PREVIOUS MODE FIELD OF PSL.
                                0027   518 ;
                                0027   519 ; OUTPUTS:
                                0027   520 ;
                                0027   521 ;       THE SPECIFIED ACCESS MODE IS MAXIMIZED WITH THE PREVIOUS MODE FIELD
                                0027   522 ;       OF THE CURRENT PSL AND RETURNED IN REGISTER R0.
                                0027   523 ;
                                0027   524 ;       REGISTERS R2 AND R3 ARE PRESERVED ACROSS CALL.
                                0027   525 ;-
                                0027   526
                           0000019A   527          .PSECT  Y$EXEPAGED
                                019A   528 EXE$MAXACMODE::                              ;MAXIMIZE ACCESS MODE
                      51    DC  019A   529          MOVPSL  R1                          ;READ CURRENT PSL
          50    51    02    16  ED  019C   530          CMPZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R0 ;COMPARE WITH PREVIOUS MODE
                      05    15  01A1   531          BLEQ    10$                         ;IF LEQ SPECIFIED ACCESS MODE LESS PRIVILEGE
          50    51    02    16  EF  01A3   532          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R0 ;EXTRACT PREVIOUS MODE FIELD
                      05        01A8   533 10$:     RSB                                 ;
```

```
                                01A9   535          .SBTTL  REMOVE ENTRY FROM TIME DEPENDENT SCHEDULER QUEUE
                                01A9   536 ;+
                                01A9   537 ; EXE$RMVTIMQ - REMOVE ENTRY FROM TIME DEPENDENT SCHEDULER QUEUE
                                01A9   538 ;
                                01A9   539 ; THIS ROUTINE IS CALLED TO REMOVE ONE OR MORE ENTRIES FROM THE TIME
                                01A9   540 ; DEPENDENT SCHEDULER QUEUE. ENTRIES ARE REMOVED BY TYPE, ACCESS MODE,
                                01A9   541 ; REQUEST IDENTIFICATION, AND PROCESS ID.
                                01A9   542 ;
                                01A9   543 ; INPUTS:
                                01A9   544 ;
                                01A9   545 ;       R2 = ACCESS MODE (ALL EQUAL AND HIGHER ACCESS MODES).
                                01A9   546 ;       R3 = REQUEST IDENTIFICATION (ZERO IMPLIES ALL).
                                01A9   547 ;       R4 = TYPE OF ENTRY TO REMOVE (ALL SINGLE AND REPEAT ENTRIES).
                                01A9   548 ;       R5 = PROCESS ID OF PROCESS TO REMOVE ENTRIES FOR.
                                01A9   549 ;
                                01A9   550 ;       IPL MUST BE IPL$_TIMER OR ABOVE.
                                01A9   551 ;
                                01A9   552 ;       IF SYSTEM SUBROUTINE OR WAKE REQUESTS ARE BEING REMOVED, THEN ACCESS
                                01A9   553 ;       MODE AND REQUEST IDENTIFICATION ARE NOT USED AND NEED NOT BE SUPPLIED
                                01A9   554 ;       IN THE CALLING SEQUENCE.
                                01A9   555 ;
                                01A9   556 ; OUTPUTS:
                                01A9   557 ;
                                01A9   558 ;       ALL ENTRIES OF THE SPECIFIED TYPE ARE REMOVED FROM THE TIME
                                01A9   559 ;       DEPENDENT SCHEDULER QUEUE.
                                01A9   560 ;-
                                01A9   561
                           00000027    562          .PSECT
                                0027   563 EXE$RMVTIMQ::                              ;REMOVE ENTRY FROM TIME QUEUE
            50    0000'CF    DE   0027   564 10$:     MOVAL   W^EXE$GL_TQFL,R0        ;GET ADDRESS OF TIMER QUEUE LISTHEAD
                  51    50   D0   002C   565          MOVL    R0,R1                   ;COPY LISTHEAD ADDRESS
                  51    61   D0   002F   566 20$:     MOVL    TQE$L_TQFL(R1),R1       ;GET ADDRESS OF NEXT ENTRY
                  50    51   D1   0032   567          CMPL    R1,R0                   ;END OF QUEUE?
                        61   13   0035   568          BEQL    80$                     ;IF EQL YES
      54    0B A1     02     00   ED   0037   569          CMPZV   #0,#2,TQE$B_RQTYPE(R1),R4 ;REQUEST TYPE MATCH?
                              F0   12   003D   570          BNEQ    20$                     ;IF NEQ NO
                  0C A1       55   D1   003F   571          CMPL    R5,TQE$L_PID(R1)        ;PROCESS ID MATCH?
                              EA   12   0043   572          BNEQ    20$                     ;IF NEQ NO
                        54    00   91   0045   573          CMPB    #TQE$C_TMSNGL,R4        ;SYSTEM SUBROUTINE OR CANCEL WAKE?
                              12   12   0048   574          BNEQ    40$                     ;IF NEQ YES - SKIP ID-ACCESS TEST
                              53   D5   004A   575          TSTL    R3                      ;ALL REQUEST ID'S MATCH?
                              06   13   004C   576          BEQL    30$                     ;IF EQL YES
                  14 A1       53   D1   004E   577          CMPL    R3,TQE$L_ASTPRM(R1)     ;REQUEST IDENTIFICATION MATCH?
                              DB   12   0052   578          BNEQ    20$                     ;IF NEQ NO
      52    28 A1     02     00   ED   0054   579 30$:     CMPZV   #0,#2,TQE$B_RMOD(R1),R2 ;REQUEST MODE LESS THAN SPECIFIED MODE?
                              D3   19   005A   580          BLSS    20$                     ;IF LSS YES
                              0C   BB   005C   581 40$:     PUSHR   #^M<R2,R3>              ;SAVE REGISTERS R2 AND R3
                     04 A1    DD   005E   582          PUSHL   TQE$L_TQBL(R1)          ;SAVE BACKWARD LINK OF ENTRY
                              50   DD   0061   583          PUSHL   R0                      ;SAVE REGISTER R0
                        50    61   0F   0063   584          REMQUE  TQE$L_TQFL(R1),R0       ;REMOVE ENTRY FROM QUEUE
                        54    01   91   0066   585          CMPB    #TQE$C_SSSNGL,R4        ;CANCEL SYSTEM SUBROUTINE?
                              29   13   0069   586          BEQL    70$                     ;IF EQL YES
                  51    2C A0     3C   006B   587          MOVZWL  TQE$L_RQPID(R0),R1      ;GET PROCESS INDEX
            51    0000'DF41       D0   006F   588          MOVL    @W^SCH$GL_PCBVEC[R1],R1 ;GET ADDRESS OF PROCESS PCB
             60 A1    2C A0     D1   0075   589          CMPL    TQE$L_RQPID(R0),PCB$L_PID(R1) ;PROCESS ID MATCH?
                              15   12   007A   590          BNEQ    60$                     ;IF NEQ NO
                        54    02   91   007C   591          CMPB    #TQE$C_WKSNGL,R4        ;CANCEL WAKE UP REQUEST?
```

```
                0D  13  007F   592          BEQL    50$                             ;IF EQL YES
          52  0080 C1  D0  0081   593          MOVL    PCB$L_JIB(R1),R2                ;GET JIB ADDRESS
                34 A2  B6  0086   594          INCW    JIB$W_TQCNT(R2)                 ;UPDATE AVAILABLE TIME QUEUE ENTRIES
       03 28 A0   06  E1  0089   595          BBC     #ACB$V_QUOTA,TQE$B_RMOD(R0),60$ ;IF CLR, NO AST SPECIFIED
                38 A1  B6  008E   596 50$:     INCW    PCB$W_ASTCNT(R1)                ;UPDATE AVAILABLE AST QUEUE ENTRIES
                  FF6C' 30  0091   597 60$:     BSBW    EXE$DEANONPAGED                 ;DEALLOCATE TIME QUEUE ENTRY
                   0F   BA  0094   598 70$:     POPR    #^M<R0,R1,R2,R3>                ;RESTORE REGISTERS R0, R1, R2, AND R3
                   97   11  0096   599          BRB     20$                             ;
                        05  0098   600 80$:     RSB                                     ;
                            0099   601
```

```
                     0099   603          .SBTTL  EXE$PROBER - Check read accessibility of user buffer
                     0099   604
                     0099   605 ;++
                     0099   606 ;
                     0099   607 ; FUNCTIONAL DESCRIPTION:
                     0099   608 ;
                     0099   609 ;       This routine performs a series of PROBEs to check the read
                     0099   610 ;       accessibility of the user-supplied buffer. Multiple PROBEs
                     0099   611 ;       must be done because the PROBE instruction only checks the
                     0099   612 ;       first and last pages while the user buffer may span several pages.
                     0099   613 ;
                     0099   614 ; CALLING SEQUENCE:
                     0099   615 ;
                     0099   616 ;       JSB/BSB EXE$PROBER
                     0099   617 ;
                     0099   618 ; INPUTS:
                     0099   619 ;
                     0099   620 ;       R0      Buffer address to be probed
                     0099   621 ;       R1      Buffer length
                     0099   622 ;       R3      Access mode to maximize with PSL<PRVMOD>
                     0099   623 ;
                     0099   624 ; SIDE EFFECTS:
                     0099   625 ;
                     0099   626 ;       R0 through R2 are destroyed
                     0099   627 ;
                     0099   628 ; ROUTINE VALUE:
                     0099   629 ;
                     0099   630 ;       R0 low bit set => successful return (SS$_NORMAL)
                     0099   631 ;       R0 low bit clear => portion of buffer is inaccessible (SS$_ACCVIO)
                     0099   632 ;
                     0099   633 ;--
                     0099   634
                00000059    635          .PSECT  AEXENONPAGED
                     0059   636
                     0059   637 EXE$PROBER::
      51    50     C0  0059   638          ADDL    R0,R1                   ; Ending address of buffer
50    01FF 8F     AA  005C   639          BICW    #VA$M_BYTE,R0           ; Truncate to start of page
      51    50     C2  0061   640          SUBL    R0,R1                   ; Calculate length of buffer to probe
52    FE00 8F     32  0064   641          CVTWL   #-^X200,R2              ; Set address adjustment constant
      51    51     F7  0069   642 10$:     CVTLW   R1,R1                   ; Greater than 32K?
            13     1D  006C   643          BVS     30$                     ; If VS, yes; check by chunks
                     006E   644 20$:
                     006E   645          IFNORD  R1,(R0),ACCVIO,R3       ; Can ends of user's buffer be read?
      50    52     C2  0074   646          SUBL    R2,R0                   ; Calculate VA of next page
   51     6142    3E  0077   647          MOVAW   (R1)[R2],R1             ; Calculate new length
            F1     14  007B   648          BGTR    20$                     ; If GTR then more to test
      50    01     3C  007D   649          MOVZWL  #SS$_NORMAL,R0          ; Indicate success
                   05  0080   650          RSB                             ;  and return
                     0081   651
                     0081   652 30$:
      7E    50     7D  0081   653          MOVQ    R0,-(SP)                ; Save current values on stack
51    7E00 8F     3C  0084   654          MOVZWL  #^X7E00,R1              ; Size of chunk used stepping thru buf.
                     0089   655                                          ;  (32K - 1 page)
      6E    51     C0  0089   656          ADDL    R1,(SP)                 ; Advance address by this amount
   04 AE    51     C2  008C   657          SUBL    R1,4(SP)                ; Decrease count
            DC     10  0090   658          BSBB    20$                     ; Probe chunk
         05 50     E9  0092   659          BLBC    R0,ACCVIO1              ; If LBC, no access
```

```
                50   8E   7D  0095   660          MOVQ    (SP)+,R0                ; Pop pre-adjusted values off stack
                     CF   11  0098   661          BRB     10$                     ; See if length now LT 32K.
                              009A   662
                              009A   663 ACCVIO1:
                5E   08   C0  009A   664          ADDL    #8,SP                   ; Clean off stack
                          05  009D   665          RSB
                              009E   666 ACCVIO:
                50   0C   3C  009E   667          MOVZWL  S^#SS$_ACCVIO,R0        ; Indicate access violation
                          05  00A1   668          RSB
                              00A2   669
```

```
                     00A2   671          .SBTTL  EXE$PROBEW - Check write accessibility of user buffer
                     00A2   672
                     00A2   673 ;++
                     00A2   674 ;
                     00A2   675 ; FUNCTIONAL DESCRIPTION:
                     00A2   676 ;
                     00A2   677 ;       This routine performs a series of PROBEs to check the write
                     00A2   678 ;       accessibility of the user-supplied buffer. Multiple PROBEs
                     00A2   679 ;       must be done because the PROBE instruction only checks the
                     00A2   680 ;       first and last pages while the user buffer may span several pages.
                     00A2   681 ;
                     00A2   682 ; CALLING SEQUENCE:
                     00A2   683 ;
                     00A2   684 ;       JSB/BSB EXE$PROBEW
                     00A2   685 ;
                     00A2   686 ; INPUTS:
                     00A2   687 ;
                     00A2   688 ;       R0      Buffer address to be probed
                     00A2   689 ;       R1      Buffer length
                     00A2   690 ;       R3      Access mode to maximize with PSL<PRVMOD>
                     00A2   691 ;
                     00A2   692 ; SIDE EFFECTS:
                     00A2   693 ;
                     00A2   694 ;       R0 through R2 are destroyed
                     00A2   695 ;
                     00A2   696 ; ROUTINE VALUE:
                     00A2   697 ;
                     00A2   698 ;       R0 low bit set => successful return (SS$_NORMAL)
                     00A2   699 ;       R0 low bit clear => portion of buffer is inaccessible (SS$_ACCVIO)
                     00A2   700 ;
                     00A2   701 ;--
                     00A2   702
                000000A2    703          .PSECT  AEXENONPAGED
                     00A2   704
                     00A2   705 EXE$PROBEW::
      51    50     C0  00A2   706          ADDL    R0,R1                   ; Ending address of buffer
50    01FF  8F     AA  00A5   707          BICW    #VA$M_BYTE,R0           ; Truncate to start of page
      51    50     C2  00AA   708          SUBL    R0,R1                   ; Calculate length of buffer to probe
52    FE00  8F     32  00AD   709          CVTWL   #-^X200,R2              ; Set address adjustment constant
      51    51     F7  00B2   710 10$:     CVTLW   R1,R1                   ; Greater than 32K?
            13     1D  00B5   711          BVS     30$                     ; If VS, yes; check by chunks
                       00B7   712
                       00B7   713 20$:
                       00B7   714          IFNOWRT R1,(R0),ACCVIO,R3       ; Can ends of user's buffer be written?
      50    52     C2  00BD   715          SUBL    R2,R0                   ; Calculate VA of next page
   51     6142   3E  00C0   716          MOVAW   (R1)[R2],R1             ; Calculate new length
            F1     14  00C4   717          BGTR    20$                     ; If GTR then more to test
      50    01     3C  00C6   718          MOVZWL  #SS$_NORMAL,R0          ; Indicate success
                   05  00C9   719          RSB                             ;  and return
                       00CA   720
                       00CA   721 30$:
      7E    50     7D  00CA   722          MOVQ    R0,-(SP)                ; Save current values on stack
51    7E00  8F     3C  00CD   723          MOVZWL  #^X7E00,R1              ; Size of chunk used stepping thru buf.
                       00D2   724                                          ;  (32K - 1 page)
      6E    51     C0  00D2   725          ADDL    R1,(SP)                 ; Advance address by this amount
   04 AE    51     C2  00D5   726          SUBL    R1,4(SP)                ; Decrease count
                   DC  10  00D9   727          BSBB    20$                     ; Probe chunk
```

```
                BC 50   E9  00DB   728          BLBC    R0,ACCVIO1              ; If LBC, no access
50              8E      7D  00DE   729          MOVQ    (SP)+,R0                ; Pop pre-adjusted values off stack
                CF      11  00E1   730          BRB     10$                     ; See if length now LT 32K.
                            00E3   731
```

```
                                00E3   733          .SBTTL  EXE$PROBER_DSC, EXE$PROBEW_DSC - Check buffer accessibility
                                00E3   734
                                00E3   735 ;++
                                00E3   736 ;
                                00E3   737 ; FUNCTIONAL DESCRIPTION:
                                00E3   738 ;
                                00E3   739 ;       Given the address of a buffer descriptor, this routine checks
                                00E3   740 ;       the accessibility of the buffer descriptor and the specified
                                00E3   741 ;       accessibility of the buffer.
                                00E3   742 ;
                                00E3   743 ;       ***** NOTE WELL *****
                                00E3   744 ;
                                00E3   745 ;       If the buffer is accessible as desired, the buffer descriptor
                                00E3   746 ;       information is returned in R1 and R2 including the high 16 bits
                                00E3   747 ;       of the first long word.  The caller MUST NOT fetch the
                                00E3   748 ;       descriptor again as that would open a protection hole.  Rather,
                                00E3   749 ;       he/she should store R1,R2 for later use in local storage.
                                00E3   750 ;       storage.
                                00E3   751 ;
                                00E3   752 ; CALLING SEQUENCE:
                                00E3   753 ;
                                00E3   754 ;       JSB/BSB EXE$PROBER_DSC
                                00E3   755 ;       JSB/BSB EXE$PROBEW_DSC
                                00E3   756 ;
                                00E3   757 ; INPUTS:
                                00E3   758 ;
                                00E3   759 ;       R1      Address of a buffer descriptor
                                00E3   760 ;
                                00E3   761 ; SIDE EFFECTS:
                                00E3   762 ;
                                00E3   763 ;       R3 is destroyed
                                00E3   764 ;
                                00E3   765 ; ROUTINE VALUE:
                                00E3   766 ;
                                00E3   767 ;       R0 low bit set => successful return (SS$_NORMAL)
                                00E3   768 ;               R1<0:15>  = size of buffer in bytes
                                00E3   769 ;               R1<16:31> = contents of the high word of the descriptor
                                00E3   770 ;               R2 = Buffer address
                                00E3   771 ;       R0 low bit clear => portion of buffer is inaccessible (SS$_ACCVIO)
                                00E3   772 ;               R1 and R2 destroyed
                                00E3   773 ;
                                00E3   774 ;--
                       000001A9 00E3   775          .PSECT  Y$EXEPAGED                      ; This code can page
                                01A9   776          .ENABL  LSB
                                01A9   777
                                01A9   778 EXE$PROBER_DSC::
                          53    D4 01A9   779          CLRL    R3                          ; Flag to indicate read probe
                          03    11 01AB   780          BRB     10$
                                01AD   781 EXE$PROBEW_DSC::
                      53    01    D0 01AD   782          MOVL    #1,R3                       ; Flag to indicate write probe
                                01B0   783 10$:       IFNORD  #8,(R1),NOACCESS            ; Error if can't read descriptor
                      50    01    D0 01B6   784          MOVL    #SS$_NORMAL,R0              ; Assume buffer is accessible
                      51    61    7D 01B9   785          MOVQ    (R1),R1                     ; R1<0:15> = size of buf, R2 = adr
                                01BC   786                                                  ; R1<16:31> = undefined,
                                01BC   787                                                  ; but must be returned to caller.
                0200  8F    51    B1 01BC   788          CMPW    R1,#512                     ; Will one probe cover the buffer?
                          11    1A 01C1   789          BGTRU   50$                         ; Branch if not, need a probe loop
```

```
                06 53   E8  01C3   790          BLBS    R3,30$                  ; Branch if checking for write access
            62  51 00   0C  01C6   791          PROBER  #0,R1,(R2)              ; See if buffer can be read by caller
               21   12   01CA   792          BNEQ    80$                     ; Branch if buffer can be read
                         01CC   793                                           ; otherwise, flow through the PROBEW
                         01CC   794                                           ; and return SS$_ACCVIO
            62  51 00   0D  01CC   795 30$:    PROBEW  #0,R1,(R2)              ; See if buffer can be written by caller
               1B   12   01D0   796          BNEQ    80$                     ; Branch if write access is allowed
               1A   11   01D2   797          BRB     NOACCESS                ; Buffer cannot be accessed
                         01D4   798 ;
                         01D4   799 ; Need to use a PROBE loop to check this buffer for the desired access.
                         01D4   800 ;
               06   BB   01D4   801 50$:    PUSHR   #^M<R1,R2>              ; Save contents of buffer descriptor
                         01D6   802                                           ; Caller must not refetch these
            0059'CF  9F  01D6   803          PUSHAB  W^EXE$PROBER            ; Assume checking for read access
         05 53   00   E5  01DA   804          BBCC    #0,R3,60$               ; Br if checking read access
                         01DE   805                                           ; and set R3 = 0
         6E 00A2'CF  9E  01DE   806          MOVAB   W^EXE$PROBEW,(SP)       ; Check for write access
            50   52   D0  01E3   807 60$:    MOVL    R2,R0                   ; Buffer address
            51   51   3C  01E6   808          MOVZWL  R1,R1                   ; Size of buffer in bytes
                 9E   16  01E9   809          JSB     @(SP)+                  ; Call PROBER/W
                 06   BA  01EB   810          POPR    #^M<R1,R2>              ; Recover contents of buf descriptor
                      05  01ED   811 80$:     RSB                             ; Return with R0 = status
                         01EE   812 NOACCESS:
            50   0C   3C  01EE   813          MOVZWL  S^#SS$_ACCVIO,R0
                      05  01F1   814          RSB
                         01F2   815
                         01F2   816          .DSABL  LSB
```

```
                                01F2   818          .SBTTL  EXE$VAL_IDNAME - Validate ID name
                                01F2   819 ;++
                                01F2   820 ;
                                01F2   821 ; EXE$VAL_IDNAME - Validate ID name
                                01F2   822 ;
                                01F2   823 ; FUNCTIONAL DESCRIPTION:
                                01F2   824 ;
                                01F2   825 ;       This routine checks the accessibility of the ID name descriptor
                                01F2   826 ;       and the name buffer. It then validates the ID name. Identifier
                                01F2   827 ;       names are 1 to 32 characters in length, consist of alpha, numeric,
                                01F2   828 ;       $, or _ characters, and must contain at least one non-numeric
                                01F2   829 ;       character.
                                01F2   830 ;
                                01F2   831 ;       ***** NOTE WELL *****
                                01F2   832 ;
                                01F2   833 ;       If the buffer is accessible as desired, the buffer descriptor
                                01F2   834 ;       information is returned in R1 and R2. The caller MUST NOT fetch
                                01F2   835 ;       the descriptor again as that would open a protection hole.  Rather,
                                01F2   836 ;       he/she should store R1,R2 for later use in local storage.
                                01F2   837 ;
                                01F2   838 ; CALLING SEQUENCE:
                                01F2   839 ;
                                01F2   840 ;       JSB/BSB EXE$VAL_IDNAME
                                01F2   841 ;
                                01F2   842 ; INPUTS:
                                01F2   843 ;
                                01F2   844 ;       R1      Address of an ID name buffer descriptor
                                01F2   845 ;
                                01F2   846 ; SIDE EFFECTS:
                                01F2   847 ;
                                01F2   848 ;       R3      destroyed
                                01F2   849 ;
                                01F2   850 ; ROUTINE VALUE:
                                01F2   851 ;
                                01F2   852 ;       R0 low bit set => successful return (SS$_NORMAL)
                                01F2   853 ;                         R1 = size of buffer in bytes
                                01F2   854 ;                         R2 = Buffer address
                                01F2   855 ;       R0 low bit clear => portion of buffer is inaccessible or an
                                01F2   856 ;                           invalid name was specified. R1 and R2 destroyed
                                01F2   857 ;
                                01F2   858 ;--
                           000001F2    859          .PSECT  Y$EXEPAGED
                                01F2   860 EXE$VAL_IDNAME::
                           FFB4 30     01F2   861          BSBW    EXE$PROBER_DSC          ; Check ID name buffer accessibility
                         2C 50 E9      01F5   862          BLBC    R0,3$                   ; Br if no access
                51  FFFF0000 8F CA     01F8   863          BICL    #^XFFFF0000,R1          ; R1 = Id name size
                         06 BB         01FF   864          PUSHR   #^M<R1,R2>              ; Save size and address
                      20 51 D1         0201   865          CMPL    R1,#KGB$S_NAME          ; Invalid name size?
                         15 1A         0204   866          BGTRU   1$                      ; Br if yes
            01  FDF4 CF 62 51 2B      0206   867          SPANC   R1,(R2),SCAN_TABLE,#MASK_M_NUMBER ; Span the numeric characters
                         0C 13         020D   868          BEQLU   1$                      ; Z bit set means id name has numerics
                                       020F   869                                          ; only. (The entire string was scanned)
                02  63  61 50 2B      020F   870          SPANC   R0,(R1),(R3),#MASK_M_ID_NAME ; Span valid characters
                         05 12         0214   871          BNEQU   1$                      ; Br if we found an invalid character
                      50 01 D0         0216   872          MOVL    #SS$_NORMAL,R0          ; Return success
                         07 11         0219   873          BRB     2$
                50  00002224 8F D0     021B   874  1$:     MOVL    #SS$_IVIDENT,R0         ; Return error
```

```
06   BA  0222   875 2$:     POPR    #^M<R1,R2>              ; Restore size and address
     05  0224   876 3$:     RSB
         0225   877
         0225   878         .END
```

```
$$                              = 00000001
ACB$V_QUOTA                     = 00000006
ACCVIO                            0000009E R      04
ACCVIO1                           0000009A R      04
ACL$W_SIZE                      = 00000008
ARM$M_DELETE                    = 00000008
ARM$M_EXECUTE                   = 00000004
ARM$M_READ                      = 00000001
ARM$M_WRITE                     = 00000002
CHAR                            = 00000100
CHAR100                         = 00000002
CHAR101                         = 00000002
CHAR102                         = 00000002
CHAR103                         = 00000002
CHAR104                         = 00000002
CHAR105                         = 00000002
CHAR106                         = 00000002
CHAR107                         = 00000002
CHAR108                         = 00000002
CHAR109                         = 00000002
CHAR110                         = 00000002
CHAR111                         = 00000002
CHAR112                         = 00000002
CHAR113                         = 00000002
CHAR114                         = 00000002
CHAR115                         = 00000002
CHAR116                         = 00000002
CHAR117                         = 00000002
CHAR118                         = 00000002
CHAR119                         = 00000002
CHAR120                         = 00000002
CHAR121                         = 00000002
CHAR122                         = 00000002
CHAR36                          = 00000002
CHAR48                          = 00000003
CHAR49                          = 00000003
CHAR50                          = 00000003
CHAR51                          = 00000003
CHAR52                          = 00000003
CHAR53                          = 00000003
CHAR54                          = 00000003
CHAR55                          = 00000003
CHAR56                          = 00000003
CHAR57                          = 00000003
CHAR65                          = 00000002
CHAR66                          = 00000002
CHAR67                          = 00000002
CHAR68                          = 00000002
CHAR69                          = 00000002
CHAR70                          = 00000002
CHAR71                          = 00000002
CHAR72                          = 00000002
CHAR73                          = 00000002
CHAR74                          = 00000002
CHAR75                          = 00000002
CHAR76                          = 00000002
CHAR77                          = 00000002
CHAR78                          = 00000002
CHAR79                          = 00000002
CHAR80                          = 00000002
CHAR81                          = 00000002
CHAR82                          = 00000002
CHAR83                          = 00000002
CHAR84                          = 00000002
CHAR85                          = 00000002
CHAR86                          = 00000002
CHAR87                          = 00000002
CHAR88                          = 00000002
CHAR89                          = 00000002
CHAR90                          = 00000002
CHAR95                          = 00000002
CHAR97                          = 00000002
CHAR98                          = 00000002
CHAR99                          = 00000002
CHP$M_READ                      = 00000001
CHP$M_USEREADALL                = 00000004
CHP$M_WRITE                     = 00000002
CHPCTL$C_LENGTH                 = 0000000C
CHPCTL$L_ACCESS                 = 00000000
CHPCTL$L_FLAGS                  = 00000004
CHPCTL$M_READ                   = 00000001
CHPCTL$M_USEREADALL             = 00000004
CHPCTL$M_WRITE                  = 00000002
CTL$GL_PCB                        ********    X   03
EXE$BUFFRQUOTA                    00000000 RG     04
EXE$BUFQUOPRC                     0000000C RG     04
EXE$CHKCREACCES                   00000105 RG     03
EXE$CHKDELACCES                   00000100 RG     03
EXE$CHKEXEACCES                   0000010F RG     03
EXE$CHKLOGACCES                   00000100 RG     03
EXE$CHKPHYACCES                   00000105 RG     03
EXE$CHKPRO_INT                    ********      X 03
EXE$CHKRDACCES                    0000010A RG     03
EXE$CHKWRTACCES                   00000114 RG     03
EXE$CLEANUP_ORB                   00000161 RG     03
EXE$DEANONPAGED                   ********      X 01
EXE$DEAPGDSIZ                     ********      X 03
EXE$GL_TQFL                       ********      X 01
EXE$INSTIMQ                       00000000 RG     01
EXE$MAXACMODE                     0000019A RG     03
EXE$MULTIQUOTA                    00000025 RG     04
EXE$PROBER                        00000059 RG     04
EXE$PROBER_DSC                    000001A9 RG     03
EXE$PROBEW                        000000A2 RG     04
EXE$PROBEW_DSC                    000001AD RG     03
EXE$RMVTIMQ                       00000027 RG     01
EXE$SNGLEQUOTA                    00000022 RG     04
EXE$VAL_IDNAME                    000001F2 RG     03
IOC$GW_MAXBUF                     ********    X   04
IPL$_ASTDEL                     = 00000002
IPL$_SYNCH                      = 00000008
JIB$L_BYTCNT                    = 00000020
JIB$L_BYTLM                     = 00000024
JIB$W_TQCNT                     = 00000034
```

```
KGB$S_NAME                      =  00000020
MASK_K_NUM                      =  00000002
MASK_M_ID_NAME                  =  00000002
MASK_M_NUMBER                   =  00000001
MASK_V_ID_NAME                  =  00000001
MASK_V_NUMBER                   =  00000000
NOACCESS                           000001EE R      03
ORB$B_FLAGS                     =  0000000B
ORB$L_ACLFL                     =  00000028
ORB$L_ACL_MUTEX                 =  00000004
ORB$V_ACL_QUEUE                 =  00000001
PCB$L_ARB                       =  0000008C
PCB$L_JIB                       =  00000080
PCB$L_PID                       =  00000060
PCB$L_STS                       =  00000024
PCB$V_SSRWAIT                   =  0000000A
PCB$W_ASTCNT                    =  00000038
PR$_IPL                         =  00000012
PSL$S_PRVMOD                    =  00000002
PSL$V_PRVMOD                    =  00000016
RSN$_ASTWAIT                    =  00000001
SCAN_TABLE                         00000000 R      03
SCH$GL_PCBVEC                      ********   X    01
SCH$LOCKW                          ********   X    03
SCH$RWAIT                          ********   X    04
SCH$UNLOCK                         ********   X    03
SS$_ACCVIO                      =  0000000C
SS$_EXQUOTA                     =  0000001C
SS$_IVIDENT                     =  00002224
SS$_NORMAL                      =  00000001
TQE$B_RMOD                      =  00000028
TQE$B_RQTYPE                    =  0000000B
TQE$C_SSSNGL                    =  00000001
TQE$C_TMSNGL                    =  00000000
TQE$C_WKSNGL                    =  00000002
TQE$L_ASTPRM                    =  00000014
TQE$L_PID                       =  0000000C
TQE$L_RQPID                     =  0000002C
TQE$L_TQBL                      =  00000004
TQE$L_TQFL                      =  00000000
TQE$Q_TIME                      =  00000018
UCB$L_ORB                       =  0000001C
VA$M_BYTE                       =  000001FF
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000099 (153.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y$EXEPAGED	00000225 (549.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
AEXENONPAGED	000000E3 (227.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.07	00:00:01.84
Command processing	153	00:00:00.60	00:00:07.23
Pass 1	495	00:00:20.93	00:01:23.13
Symbol table sort	0	00:00:02.64	00:00:11.50
Pass 2	170	00:00:04.24	00:00:13.70
Symbol table output	19	00:00:00.16	00:00:00.90
Psect synopsis output	2	00:00:00.02	00:00:00.30
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	876	00:00:28.67	00:01:58.60

The working set limit was 1800 pages.
121151 bytes (237 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1706 non-local and 36 local symbols.
878 source lines were read in Pass 1, producing 18 object records in Pass 2.
41 pages of virtual memory were used to define 39 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	17
_255DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	31

1783 GETS were required to define 31 macros.

There were no errors, warnings or information messages.

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

FILEREAD
LIS

EXCEPTMSG
LIS

DISMOUNT
LIS

DEBUGDATA
LIS

ERRORLOG
LIS

DEVICEDAT
LIS

EXCEPTION
LIS

FILERWIO
LIS

EXSUBRÔUT
LIS