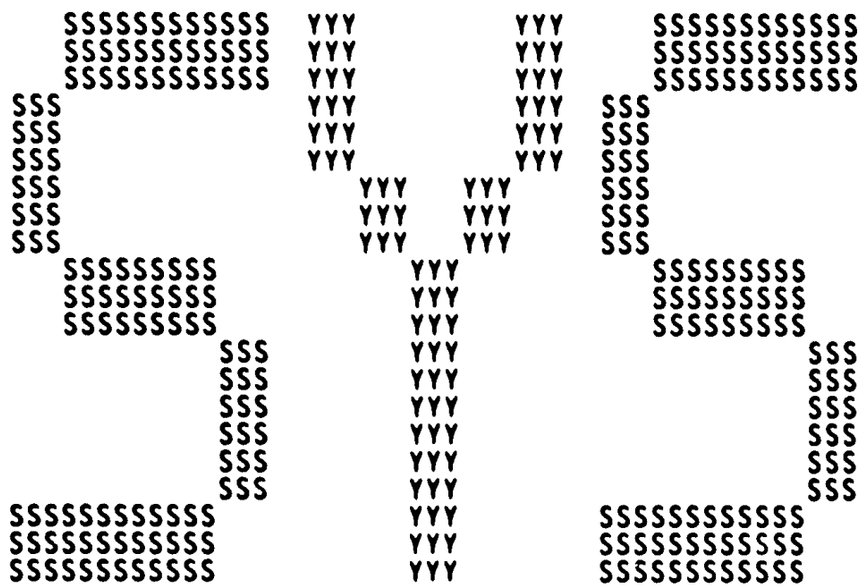

```
MM       MM  EEEEEEEEEE  MM       MM    000000    RRRRRRRR    YY       YY    AAAAAA    LL            CCCCCCCC
MM       MM  EEEEEEEEEE  MM       MM    000000    RRRRRRRR    YY       YY    AAAAAA    LL            CCCCCCCC
MMMM   MMMM  EE          MMMM   MMMM  00      00  RR      RR  YY       YY  AA      AA  LL          CC
MMMM   MMMM  EE          MMMM   MMMM  00      00  RR      RR  YY       YY  AA      AA  LL          CC
MM  MM  MM  EE          MM  MM  MM  00      00  RR      RR    YY  YY    AA      AA  LL          CC
MM  MM  MM  EE          MM  MM  MM  00      00  RR      RR    YY  YY    AA      AA  LL          CC
MM       MM  EEEEEEEE    MM       MM  00      00  RRRRRRRR        YY    AA      AA  LL          CC
MM       MM  EEEEEEEE    MM       MM  00      00  RRRRRRRR        YY    AA      AA  LL          CC
MM       MM  EE          MM       MM  00      00  RR  RR          YY    AAAAAAAAAA  LL          CC
MM       MM  EE          MM       MM  00      00  RR  RR          YY    AAAAAAAAAA  LL          CC
MM       MM  EE          MM       MM  00      00  RR    RR        YY    AA      AA  LL          CC                ....
MM       MM  EE          MM       MM  00      00  RR    RR        YY    AA      AA  LL          CC                ....
MM       MM  EEEEEEEEEE  MM       MM    000000    RR      RR      YY    AA      AA  LLLLLLLLLL    CCCCCCCC      ....
MM       MM  EEEEEEEEEE  MM       MM    000000    RR      RR      YY    AA      AA  LLLLLLLLLL    CCCCCCCC      ....

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

```
0000     1          .TITLE  MEMORYALC - DYNAMIC MEMORY ALLOCATION
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  ; D. N. CUTLER 3-AUG-76
0000    29  ;
0000    30  ; MODIFIED BY:
0000    31  ;
0000    32  ;       V03-031 WMC0011         Wayne Cardoza           27-Aug-1984
0000    33  ;               Return correct status after pool expansion fails.
0000    34  ;
0000    35  ;       V03-030 WMC0010         Wayne Cardoza           23-Aug-1984
0000    36  ;               Output message on pool expansion failure.
0000    37  ;
0000    38  ;       V03-029 DAS0002         David Solomon           30-Apr-1984
0000    39  ;               Fix broken branch destination.
0000    40  ;
0000    41  ;       V03-028 DAS0001         David Solomon           30-Apr-1984
0000    42  ;               Fix broken word reference to IOC$GL_xxx.
0000    43  ;
0000    44  ;       V03-027 CWH3027         CW Hobbs                29-Apr-1984
0000    45  ;               Fix broken branch.
0000    46  ;
0000    47  ;       V03-026 EMD0072         Ellen M. Dusseault      6-Apr-1984
0000    48  ;               Fix broken branches.
0000    49  ;
0000    50  ;       V03-025 KPL0003         Peter Lieberwirth       31-Mar-1984
0000    51  ;               Fix register useage bug in EXE$ALOPHYCNTG.  Fix some
0000    52  ;               broken branches.
0000    53  ;
0000    54  ;       V03-024 SRB0118         Steve Beckhart          26-Mar-1984
0000    55  ;               Fixed broken branches.
0000    56  ;
0000    57  ;       V03-024 KDM0095         Kathleen D. Morse       23-Mar-1984
```

```
0000     58 ;                Maximize size of slave CEB with regular CEB in EXE$ALLOCCEB.
0000     59 ;
0000     60 ;        V03-023 WMC0009         Wayne Cardoza           08-Mar-1984
0000     61 ;                Add entry point for non-paged variable length pool.
0000     62 ;
0000     63 ;        V03-022 WMC0008         Wayne Cardoza           25-Feb-1984
0000     64 ;                Optimize EXE$ALLOCBUF
0000     65 ;
0000     66 ;        V03-021 KPL0002         Peter Lieberwirth       22-Feb-1984
0000     67 ;                Fix bugs in V03-019, specifically: move routine to
0000     68 ;                non-paged PSECT, correct pointers that are zero-based.
0000     69 ;
0000     70 ;        V03-020 LY00B3          Larry Yetto             10-FEB-1984 10:10
0000     71 ;                Fix truncation errors
0000     72 ;
0000     73 ;        V03-019 KPL0001         Peter Lieberwirth        5-Feb-1984
0000     74 ;                Add EXE$ALOPHYCNTG routine that allocates physically
0000     75 ;                contiguous memory and maps it in system virtual address
0000     76 ;                space.
0000     77 ;
0000     78 ;        V03-018 LJK0257         Lawrence J. Kenah       6-Jan-1984
0000     79 ;                Add EXE$ALOPAGWAIT routine that parallels a similar for
0000     80 ;                nonpaged pool. Remove EXE$ALLOCPQB entry point.
0000     81 ;
0000     82 ;        V03-017 DWT0157         David W. Thiel          29-Dec-1983
0000     83 ;                Fix broken branches.
0000     84 ;
0000     85 ;        V03-016 TMK0001         Todd M. Katz            13-Nov-1983
0000     86 ;                Fix a broken branch.
0000     87 ;
0000     88 ;        V03-015 CWH3015         CW Hobbs                29-Oct-1983
0000     89 ;                Fix broken branch.
0000     90 ;
0000     91 ;        V03-014 DWT0123         David W. Thiel          23-Aug-1983
0000     92 ;                Improve performance of main path through EXE$ALLOCATE by
0000     93 ;                deleting the calculation of the largest free block.
0000     94 ;                Improve performance of EXE$DEANONPGDSIZ by avoiding
0000     95 ;                call to SCH$RAVAIL on returning a block to a lookaside
0000     96 ;                list when the lookaside list was not previously empty.
0000     97 ;                Add EXE$DEAPGDSIZ entry point to deallocate an
0000     98 ;                arbirtrary quantity of paged dynamic memory.
0000     99 ;
0000    100 ;        V03-013 SRB0098         Steve Beckhardt         19-Jul-1983
0000    101 ;                Fixed bug in extending pool.  Changed EXE$ALONONPAGED
0000    102 ;                to return SS$_INSFMEM on error.
0000    103 ;
0000    104 ;        V03-012 SRB0076         Steve Beckhardt         8-Apr-1983
0000    105 ;                Modified EXE$ALONONPAGED to preserve R3.  Added entry point
0000    106 ;                EXE$ALONPAGWAITS which is similar to EXE$ALONPAGWAIT but
0000    107 ;                may be called at IPL$_SYNCH.
0000    108 ;
0000    109 ;        V03-011 ROW0178         Ralph O. Weber          5-APR-1983
0000    110 ;                Enhance the "cannot expand pool right now because IPL is
0000    111 ;                wrong" test in EXE$EXTENDPOOL to allow pool expansion when the
0000    112 ;                current IPL is at or below IPL$_SYNCH or the processor is not
0000    113 ;                running on the interrupt stack.
0000    114 ;
```

```
0000    115 ;       V03-010 RSH0006         R. Scott Hanna          28-Feb-1983
0000    116 ;               Temporary disable of size and address granularity check
0000    117 ;               in EXE$DEAP1. This check should be restored when all users
0000    118 ;               of the P1 allocation region call EXE$ALOP1PROC or
0000    119 ;               EXE$ALPO1IMAG to allocate space.
0000    120 ;
0000    121 ;       V03-009 JWH0183         Jeffrey W. Horn         10-Feb-1983
0000    122 ;               Modify P1 routines to round requests to proper allocation
0000    123 ;               boundries.
0000    124 ;
0000    125 ;       V03-008 STJ3052         Steven Jeffreys         20-Jan-1983
0000    126 ;               Changed W^ to L^ to fix link truncation error.
0000    127 ;
0000    128 ;       V03-007 WMC0007         Wayne Cardoza           10-Jan-1983
0000    129 ;               Fix MOVZBL of PCB length (grew past byte length).
0000    130 ;
0000    131 ;       V03-006 JWH0142         Jeffrey W. Horn         20-Nov-1982
0000    132 ;               Optimize routines added in JWH0119.  Also allow these
0000    133 ;               routines to be called from elevated IPL.
0000    134 ;
0000    135 ;       V03-005 JWH0119         Jeffrey W. Horn          3-Nov-1982
0000    136 ;               Add  EXE$ALOP1IMAG, EXE$ALOP1PROC, and EXE$DEAP1
0000    137 ;               routines.
0000    138 ;
0000    139 ;       V03-004 CWH0001         CW Hobbs                19-Aug-1982
0000    140 ;               Add call to $PFNDEF macro
0000    141 ;
0000    142 ;       V03-003 WMC0001         Wayne Cardoza           31-Jul-1982
0000    143 ;               Fill in PFN data base when expanding non-paged pool.
0000    144 ;
0000    145 ;       V03-002 SRB0052         Steve Beckhardt         2-Jun-1982
0000    146 ;               Modified EXE$DEALLOCATE to detect overlapping deallocates
0000    147 ;               and bugcheck if this occurs.
0000    148 ;
0000    149 ; DYNAMIC MEMORY ALLOCATION
0000    150 ;
0000    151 ; MACRO LIBRARY CALLS
0000    152 ;
0000    153
0000    154         $CEBDEF                         ;DEFINE COMMON EVENT BLOCKS
0000    155         $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
0000    156         $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
0000    157         $IRPDEF                         ;DEFINE IRP OFFSETS
0000    158         $IMPDEF                         ;DEFINE RMS IMPURE AREA SYMBOLS
0000    159         $JIBDEF                         ;DEFINE JIB OFFSETS
0000    160         $PCBDEF                         ;DEFINE PCB OFFSETS
0000    161         $PFNDEF                         ;DEFINE PFN CONSTANTS & OFFSETS
0000    162         $PHDDEF                         ;DEFINE PHD OFFSETS
0000    163         $PQBDEF                         ;DEFINE PQB OFFSETS
0000    164         $PRDEF                          ;DEFINE PROCESSOR REGISTERS
0000    165         $PSLDEF                         ;DEFINE PSL FIELDS
0000    166         $PTEDEF                         ;DEFINE PTE FIELDS
0000    167         $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
0000    168         $SHBDEF                         ;DEFINE SHARED MEM CONTROL BLOCK
0000    169         $SHDDEF                         ;DEFINE SHARED MEM DATAPAGE
0000    170         $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
0000    171         $TQEDEF                         ;DEFINE TQE OFFSETS
```

```
                     0000    172          $TTYDEF                         ;TWP
                     0000    173          $VADEF                          ;DEFINE VIRTUAL ADDRESS FIELDS
                     0000    174
                     0000    175 ;
                     0000    176 ; LOCAL SYMBOLS
                     0000    177 ;
                     0000    178 ; ALLOCATION GRANULARITY MASK
                     0000    179 ;
                     0000    180
0000000F             0000    181 MASK=^XF                                 ;16 BYTE ALLOCATION GRANULARITY
0000000F             0000    182 EXE$C_ALCGRNMSK==MASK                    ; ALLOCATION GRANULARITY MASK
                     0000    183
                00000000     184          .PSECT  AEXENONPAGED            ; NONPAGED EXEC
```

```
0000    186          .SBTTL  ALLOCATE MEMORY AND CONDITIONALLY WAIT
0000    187 ;+
0000    188 ; EXE$ALLOCBUF - ALLOCATE BUFFERED I/O BUFFER AND CONDITIONALLY WAIT
0000    189 ;
0000    190 ; THIS ROUTINE IS CALLED TO ALLOCATE A BUFFERED I/O BUFFER. IF SUFFICIENT
0000    191 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
0000    192 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000    193 ;
0000    194 ; EXE$ALLOCCEB - ALLOCATE COMMON EVENT BLOCK AND CONDITIONALLY WAIT
0000    195 ;
0000    196 ; THIS ROUTINE IS CALLED TO ALLOCATE A COMMON EVENT BLOCK. IF SUFFICIENT
0000    197 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
0000    198 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000    199 ;
0000    200 ; EXE$ALLOCJIB - ALLOCATE JOB INFORMATION BLOCK AND CONDITIONALLY WAIT
0000    201 ;
0000    202 ; THIS ROUTINE IS CALLED TO ALLOCATE A JOB INFORMATION BLOCK. IF SUFFICIENT
0000    203 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED
0000    204 ; DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000    205 ;
0000    206 ; EXE$ALLOCIRP - ALLOCATE I/O REQUEST PACKET AND CONDITIONALLY WAIT
0000    207 ;
0000    208 ; THIS ROUTINE IS CALLED TO ALLOCATE AN I/O PACKET. IF SUFFICIENT MEMORY
0000    209 ; IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED
0000    210 ; DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000    211 ;
0000    212 ; EXE$ALLOCPCB - ALLOCATE PROCESS CONTROL BLOCK AND CONDITIONALLY WAIT
0000    213 ;
0000    214 ; THIS ROUTINE IS CALLED TO ALLOCATE A PROCESS CONTROL BLOCK WHEN
0000    215 ; CREATING A NEW PROCESS.  IF SUFFICIENT MEMORY IS NOT AVAILABLE, THEN
0000    216 ; A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED DEPENDING ON THE CURRENT
0000    217 ; PROCESS' RESOURCE WAIT MODE.
0000    218 ;
0000    219 ; EXE$ALLOCPQB - ALLOCATE PROCESS QUOTA BLOCK AND CONDITIONALLY WAIT
0000    220 ;
0000    221 ; THIS ROUTINE IS CALLED TO ALLOCATE A PROCESS QUOTA BLOCK WHEN CREATING
0000    222 ; A NEW PROCESS.  IF SUFFICIENT MEMORY IS NOT AVAILABLE, THEN A RESOURCE
0000    223 ; WAIT STATE IS ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT
0000    224 ; MODE.
0000    225 ;
0000    226 ; EXE$ALLOCTQE - ALLOCATE TIME QUEUE ENTRY AND CONDITIONALLY WAIT
0000    227 ;
0000    228 ; THIS ROUTINE IS CALLED TO ALLOCATE A TIME QUEUE ENTRY. IF SUFFICIENT
0000    229 ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
0000    230 ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000    231 ;
0000    232 ; INPUTS:
0000    233 ;
0000    234 ;       R4 = NORMALLY CURRENT PROCESS PCB ADDRESS, BUT NOT REQUIRED.
0000    235 ;
0000    236 ; IF ENTRY AT EXE$ALLOCBUF, THEN
0000    237 ;
0000    238 ;       R1 = SIZE OF REQUESTED BUFFER IN BYTES.
0000    239 ;
0000    240 ; OUTPUTS:
0000    241 ;
0000    242 ;       R0 = LOW BIT CLEAR IF ALLOCATION FAILURE WITH CALLING IPL PRESERVED.
```

```
                     0000   243 ;
                     0000   244 ;                   R0 = SS$_INSFMEM = INSUFFICIENT MEMORY AVAILABLE TO ALLOCATE
                     0000   245 ;                        BUFFER.
                     0000   246 ;
                     0000   247 ;           R0 = LOW BIT SET IF SUCCESSFUL ALLOCATION WITH:
                     0000   248 ;
                     0000   249 ;                   R1 = SIZE OF REQUESTED BUFFER IN BYTES.
                     0000   250 ;                   R2 = ADDRESS OF ALLOCATED BUFFER WITH SIZE AND TYPE FIELDS
                     0000   251 ;                        FILLED IN.
                     0000   252 ;
                     0000   253 ;                   AND IPL SET TO AST DELIVERY LEVEL.
                     0000   254 ;
                     0000   255 ;           R4 = ORIGINAL R4 OR CURRENT PCB IF A WAIT OCCURRED.
                     0000   256 ;-
                     0000   257
                     0000   258                   .ENABL  LSB
                     0000   259 EXE$ALLOCCEB::                              ;ALLOCATE COMMON EVENT BLOCK
             04   DD 0000   260                   PUSHL   #DYN$C_CEB          ;SET DATA STRUCTURE TYPE
                     0002   261                   ASSUME  CEB$C_SLAVLNG GT CEB$C_LENGTH
     51 0050 8F   3C 0002   262                   MOVZWL  #<CEB$C_SLAVLNG+MASK>&<^C<MASK>>,R1 ;AND LENGTH OF BLOCK
             1F   11 0007   263                   BRB     20$                 ;
                     0009   264 EXE$ALLOCJIB::                              ;ALLOCATE JOB INFORMATION BLOCK - COND WAIT
             2F   DD 0009   265                   PUSHL   #DYN$C_JIB          ;SET STRUCTURE TYPE
     51 0080 8F   3C 000B   266                   MOVZWL  #<JIB$C_LENGTH+MASK>&<^C<MASK>>,R1 ;AND LENGTH OF BLOCK
             16   11 0010   267                   BRB     20$                 ;MERGE WITH COMMON ALLOCATE CODE
                     0012   268 EXE$ALLOCIRP::                              ;ALLOCATE I/O PACKET - CONDITIONAL WAIT
             0A   DD 0012   269                   PUSHL   #DYN$C_IRP          ;SET DATA STRUCTURE TYPE
       51    D0 8F   9A 0014   270                   MOVZBL  #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R1 ;SET SIZE OF BUFFER REQUIRED
             0E   11 0018   271                   BRB     20$                 ;
                     001A   272 EXE$ALLOCPCB::                              ;ALLOCATE PROCESS CONTROL BLOCK
             0C   DD 001A   273                   PUSHL   #DYN$C_PCB          ;SET DATA STRUCTURE TYPE
      51    0120 8F   3C 001C   274                   MOVZWL  #<PCB$C_LENGTH+MASK>&<^C<MASK>>,R1 ;AND STRUCTURE SIZE
             05   11 0021   275                   BRB     20$                 ;
                     0023   276 EXE$ALLOCTQE::                              ;ALLOCATE TIME QUEUE ENTRY
             0F   DD 0023   277                   PUSHL   #DYN$C_TQE          ;SET DATA STRUCTURE TYPE
          51 30   9A 0025   278                   MOVZBL  #<TQE$C_LENGTH+MASK>&<^C<MASK>>,R1 ;SET SIZE OF BUFFER REQUIRED
             7E   DC 0028   279 20$:      MOVPSL  -(SP)                       ;READ CURRENT PSL
                     002A   280                   DSBINT  #IPL$_SYNCH         ;SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
             51   DD 0030   281                   PUSHL   R1                  ;SAVE REQUEST SIZE
           00B8   30 0032   282                   BSBW    EXE$ALONONPAGED     ;ATTEMPT TO ALLOCATE PACKET
             0A   BA 0035   283                   POPR    #^M<R1,R3>          ;RETRIEVE REQUEST SIZE AND PREVIOUS IPL
          13 50   E9 0037   284                   BLBC    R0,40$              ;IF LBC NO PACKET ALLOCATED
    08 A2    51   B0 003A   285                   MOVW    R1,IRP$W_SIZE(R2)   ;INSERT SIZE OF ALLOCATED BLOCK
 0A A2    04 AE   9B 003E   286                   MOVZBW  4(SP),IRP$B_TYPE(R2) ;INSERT DATA STRUCTURE TYPE
                     0043   287                                               ;AND CLEAR MISCELLANEOUS BYTE
          53   02   3C 0043   288                   MOVZWL  #IPL$_ASTDEL,R3     ;SET TO RAISE TO AST DELIVERY LEVEL
                     0046   289 30$:      ENBINT  R3                          ;ALLOW INTERRUPTS
          5E   08   C0 0049   290                   ADDL    #8,SP               ;REMOVE PSL AND STRUCTURE TYPE FROM STACK
             05   004C   291                   RSB                         ;
54   00000000'EF   D0 004D   292 40$:      MOVL    L^SCH$GL_CURPCB,R4          ;FORCE CURRENT PCB ADDRESS
   ED 24 A4   0A   E0 0054   293                   BBS     #PCB$V_SSRWAIT,PCB$L_STS(R4),30$ ;IF SET, NO WAIT
          50   03   3C 0059   294                   MOVZWL  #RSN$_NPDYNMEM,R0   ;SET NONPAGED DYNAMIC MEMORY RESOURCE NUMBER
            FFA1'  30 005C   295                   BSBW    SCH$RWAIT           ;WAIT FOR NONPAGED MEMORY
             C7   11 005F   296                   BRB     20$                 ;
                     0061   297 ;
                     0061   298 ;
                     0061   299 ; This routine is optimized due to heavy useage
```

```
                                0061   300 ;
                                0061   301 EXE$ALLOCBUF::                            ;ALLOCATE BUFFERED I/O BUFFER
                      7E   DC   0061   302          MOVPSL  -(SP)                     ;READ CURRENT PSL
                   12 08   DA   0063   303          MTPR    #IPL$_SYNCH,#PR$_IPL      ;SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
                   53 51   D0   0066   304          MOVL    R1,R3                     ;SAVE REQUEST SIZE
                      0081 30   0069   305          BSBW    EXE$ALONONPAGED           ;ATTEMPT TO ALLOCATE PACKET
                   51 53   D0   006C   306          MOVL    R3,R1                     ;RETRIEVE REQUEST SIZE
                     0F 50 E9   006F   307          BLBC    R0,50$                    ;IF LBC NO PACKET ALLOCATED
                08 A2   51   B0   0072   308          MOVW    R1,IRP$W_SIZE(R2)         ;INSERT SIZE OF ALLOCATED BLOCK
                0A A2   13   B0   0076   309          MOVW    #DYN$C_BUFIO,IRP$B_TYPE(R2) ;INSERT DATA STRUCTURE TYPE
                                007A   310                                            ;AND CLEAR MISCELLANEOUS BYTE
                   12   02   DA   007A   311          MTPR    #IPL$_ASTDEL,#PR$_IPL     ;ALLOW INTERRUPTS
                   5E   04   C0   007D   312          ADDL    #4,SP                     ;REMOVE PSL FROM STACK
                             05   0080   313          RSB
                                0081   314 ;
                   53   8E   D0   0081   315 50$:     MOVL    (SP)+,R3                  ;TEMPORARILY RETRIEVE PSL
                        13   DD   0084   316          PUSHL   #DYN$C_BUFIO              ;CONDITIONS FOR GENERAL ROUTINE
                        53   DD   0086   317          PUSHL   R3                        ;PUT PSL BACK ON STACK
53   53   F0 8F   78   0088   318          ASHL    #-PSL$V_IPL,R3,R3         ;IPL
                        BE   11   008D   319          BRB     40$                       ;CONTINUE THE SLOW WAY
                                008F   320
                                008F   321          .DSABL  LSB
```

```
                        .SBTTL  EXE$ALONPAGWAIT - ALLOCATE MEMORY AND WAIT IN CALLER'S MODE
                ;+
                ; EXE$ALONPAGWAIT - ALLOCATE NON-PAGED MEMORY AND OPTIONALLY WAIT IN CALLER'S
                ;                   ACCESS MODE.
                ;
                ; THIS ROUTINE IS CALLED TO ALLOCATE NON-PAGED MEMORY.  IF SUFFICIENT
                ; MEMORY IS NOT AVAILABLE, THEN THE PROCESS IS PLACED IN A RESOURCE
                ; WAIT STATE IF RESOURCE WAIT MODE IS ENABLED.  OTHERWISE, AN ERROR IS
                ; RETURNED.  THIS ROUTINE IS INTENDED TO BE CALLED BY SYSTEM SERVICES
                ; THAT NEED TO ALLOCATE NON-PAGED POOL.  IF IT IS NECESSARY TO WAIT, THE
                ; STACK IS TRIMMED BACK TO WAIT IN THE MODE OF THE (SYSTEM SERVICE) CALLER.
                ; WHEN THE PROCESS IS RESUMED, IT RESUMES AT THE BEGINNING OF THE SYSTEM
                ; SERVICE.  AS A RESULT OF THIS INTERFACE, IT IS NECESSARY TO PROVIDE A
                ; WAY FOR THE SYSTEM SERVICE TO PERFORM ANY NECESSARY CLEANUP BEFORE
                ; BEING PLACED IN THE WAIT STATE.  THIS IS PROVIDED BY ALLOWING THE
                ; SYSTEM SERVICE TO SPECIFY AN OPTIONAL CLEANUP ROUTINE TO BE CALLED
                ; PRIOR TO PLACING THE PROCESS IN A WAIT STATE.
                ;
                ; INPUTS:
                ;
                ;       R0 = ADDRESS OF CLEANUP ROUTINE (0 INDICATES NO ROUTINE)
                ;       R1 = SIZE OF BUFFER REQUIRED (IN BYTES)
                ;
                ;       IPL MUST BE AT IPL$_ASTDEL OR LOWER (ENTRY POINT EXE$ALONPAGWAIT)
                ;       IPL MUST BE AT IPL$_SYNCH (ENTRY POINT EXE$ALONPAGWAITS)
                ;
                ; OUTPUTS:
                ;
                ;       R0 = SS$_INSFMEM IF ALLOCATION FAILURE AND RESOURCE WAIT MODE
                ;            DISABLED.  IPL IS AT IPL$_ASTDEL (BOTH ENTRY POINTS).
                ;
                ;          = LOW BIT SET IF SUCCESSFUL ALLOCATION.
                ;            IPL IS AT IPL$_ASTDEL (ENTRY POINT EXE$ALONPAGWAIT).
                ;            IPL IS AT IPL$_SYNCH (ENTRY POINT EXE$ALONPAGWAITS).
                ;
                ;       R1 = SIZE OF ALLOCATED BUFFER
                ;
                ;       R2 = ADDRESS OF ALLOCATED BUFFER
                ;
                ;       R4 = ORIGINAL R4 IF ALLOCATION WAS SUCCESSFUL.
                ;
                ;          = CURRENT PROCESS PCB ADDRESS IF ALLOCATION FAILURE AND RESOURCE
                ;            WAIT MODE DISABLED.
                ;
                ; NOTES:
                ;
                ;       THE CLEANUP ROUTINE IS CALLED AT IPL$_SYNCH (THEREFORE IT
                ;       MUST BE NON-PAGEABLE).
                ;-
                
                        .ENABL  LSB
                
                EXE$ALONPAGWAITS::
    50    DD    008F    376         PUSHL   R0                      ; SAVE ADDRESS OF CLEANUP ROUTINE
    5A    10    0091    377         BSBB    EXE$ALONONPAGED         ; TRY TO ALLOCATE THE MEMORY
02  50    E8    0093    378         BLBS    R0,10$                  ; SKIP NEXT IF SUCCESSFUL
    29    10    0096    379         BSBB    RWAIT_CHECK_NP          ; CHECK FOR RESOURCE WAIT
```

```
                     5E   04   C0  0098   380 10$:     ADDL    #4,SP                       ; REMOVE ADDRESS OF CLEANUP ROUTINE
                          05   009B  381             RSB                                 ; RETURN
                               009C  382
                               009C  383 EXE$ALONPAGWAIT::
                          50   DD   009C  384             PUSHL   R0                          ; SAVE ADDRESS OF CLEANUP ROUTINE
                               009E  385             SETIPL  #IPL$_SYNCH                 ; RAISE IPL TO SYNCH
                          4A   10   00A1  386             BSBB    EXE$ALONONPAGED             ; TRY TO ALLOCATE THE MEMORY
                        02 50   E8   00A3  387             BLBS    R0,20$                      ; FAILED TO ALLOCATE IT
                          19   10   00A6  388             BSBB    RWAIT_CHECK_NP              ; CHECK FOR RESOURCE WAIT
                               00A8  389 20$:     SETIPL  #IPL$_ASTDEL                ; LOWER IPL TO ASTDEL
                     5E   04   C0  00AB  390             ADDL    #4,SP                       ; REMOVE ADDRESS OF CLEANUP ROUTINE
                          05   00AE  391             RSB                                 ; RETURN
                               00AF  392
                               00AF  393 EXE$ALOPAGWAIT::
                          50   DD   00AF  394             PUSHL   R0                          ; SAVE ADDRESS OF CLEANUP ROUTINE
                       00CC   30   00B1  395             BSBW    EXE$ALOPAGED                ; TRY TO ALLOCATE THE MEMORY
                        02 50   E8   00B4  396             BLBS    R0,30$                      ; FAILED TO ALLOCATE IT
                          04   10   00B7  397             BSBB    RWAIT_CHECK_PAG             ; CHECK FOR RESOURCE WAIT
                     5E   04   C0  00B9  398 30$:     ADDL    #4,SP                       ; REMOVE ADDRESS OF CLEANUP ROUTINE
                          05   00BC  399             RSB                                 ; RETURN
                               00BD  400
                               00BD  401 ; ALLOCATION FAILED. IF RESOURCE WAIT MODE IS ENABLED THEN REMOVE CALL
                               00BD  402 ; FRAME FROM STACK AND WAIT IN CALLER'S MODE. OTHERWISE, RETURN ERROR.
                               00BD  403
                               00BD  404 RWAIT_CHECK_PAG:
                          05   DD   00BD  405             PUSHL   #RSN$_PGDYNMEM              ; STORE RESOURCE NUMBER
                          02   11   00BF  406             BRB     40$                         ; JOIN COMMON CODE
                               00C1  407
                               00C1  408 RWAIT_CHECK_NP:
                               00C1  409
                          03   DD   00C1  410             PUSHL   #RSN$_NPDYNMEM              ; STORE RESOURCE NUMBER
                54 00000000'EF   D0   00C3  411 40$:     MOVL    L^SCH$GL_CURPCB,R4          ; GET CURRENT PROCESS PCB ADDRESS
                   EA 24 A4   0A   E0   00CA  412             BBS     #PCB$V_SSRWAIT,PCB$L_STS(R4),30$  ; BRANCH IF NOT WAITING
                      50   08 AE   D0   00CF  413             MOVL    8(SP),R0                    ; GET ADDRESS OF CLEANUP ROUTINE
                          02   13   00D3  414             BEQL    50$                         ; NO CLEANUP ROUTINE
                          60   16   00D5  415             JSB     (R0)                        ; CALL CLEANUP ROUTINE
                          50 8ED0   00D7  416 50$:     POPL    R0                          ; GET RESOURCE NUMBER
                        5E   5D   D0   00DA  417             MOVL    FP,SP                       ; TRIM STACK BACK TO START OF FRAME
                     5C   08 AE   7D   00DD  418             MOVQ    8(SP),AP                    ; RESTORE PRE-CALL AP AND FP
                        5E   00'  C0   00E1  419             ADDL    S^#EXE$C_CMSTKSZ,SP         ; CLEAN CALL FRAME OFF STACK
                        6E   04   C2   00E4  420             SUBL    #4,(SP)                     ; BACK UP SAVED PC TO POINT TO CHMK
                           FF16'  31   00E7  421             BRW     SCH$RWAIT                   ; WAIT FOR RESOURCE
                                  00EA  422
                                  00EA  423             .DSABL  LSB
```

```
                                00EA   425          .SBTTL  ALLOCATE NONPAGED DYNAMIC MEMORY
                                00EA   426 ;+
                                00EA   427 ; EXE$ALONONPAGED - ALLOCATE NONPAGED DYNAMIC MEMORY
                                00EA   428 ;
                                00EA   429 ; EXE$ALONPAGVAR  - ALLOCATE NONPAGED DYNAMIC MEMORY FROM VARIABLE AREA ONLY
                                00EA   430 ;
                                00EA   431 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE NONPAGED POOL.
                                00EA   432 ; IF THE BLOCK IS THE SAME SIZE AS AN I/O PACKET, AN ATTEMPT IS MADE TO ALLO-
                                00EA   433 ; CATE IT FROM THE LOOKASIDE LIST.
                                00EA   434 ;
                                00EA   435 ; INPUTS:
                                00EA   436 ;
                                00EA   437 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                                00EA   438 ;
                                00EA   439 ; OUTPUTS:
                                00EA   440 ;
                                00EA   441 ;       R0 = SS$_INSFMEM IF MEMORY IS NOT AVAILABLE.
                                00EA   442 ;
                                00EA   443 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                                00EA   444 ;
                                00EA   445 ;               R1 = SIZE OF ALLOCATED BLOCK.
                                00EA   446 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                                00EA   447 ;
                                00EA   448 ;       OTHER REGISTERS PRESERVED
                                00EA   449 ;-
                                00EA   450
                                00EA   451          .ENABL  LSB
                     00C6   31  00EA   452 200$:    BRW     20$                     ;BAD ALLOCATION REQUEST
                                00ED   453 EXE$ALONONPAGED::                       ;ALLOCATE NONPAGED MEMORY
51   000000D0 8F     D1  00ED   454          CMPL    #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R1 ;SIZE GREATER THAN IRP ?
                  16   1F  00F4   455          BLSSU   LRP                     ;IF LSSU, YES
00000000'EF       51   D1  00F6   456          CMPL    R1, L^IOC$GL_IRPMIN     ;IS THE BLOCK TOO SMALL?
                  64   1F  00FD   457          BLSSU   SRP                     ;YES, TRY SMALL PACKETS
52   00000000'FF     0F  00FF   458          REMQUE  @L^IOC$GL_IRPFL,R2      ;REMOVE FIRST PACKET FROM LOOK ASIDE LIST
                  23   1D  0106   459          BVS     LISTCHK                 ;IF VS EMPTY LIST
               50 01   D0  0108   460          MOVL    #SS$_NORMAL,R0          ;SET SUCCESSFUL COMPLETION
                        05  010B   461          RSB                             ;
                            010C   462
51   00000000'EF     D1  010C   463 LRP:       CMPL    L^IOC$GL_LRPMIN,R1      ;SIZE LESS THAN LRP MINIMUM ?
                  1C   1A  0113   464          BGTRU   EXE$ALONPAGVAR          ;IF GTRU, YES
51   00000000'EF     D1  0115   465          CMPL    L^IOC$GL_LRPSIZE,R1     ;SIZE GREATER THAN LRP ?
                  13   1F  011C   466          BLSSU   EXE$ALONPAGVAR          ;IF LSSU, YES
52   00000000'FF     0F  011E   467          REMQUE  @L^IOC$GL_LRPFL,R2      ;REMOVE FIRST PACEKT FROM LRP LIST
                  04   1D  0125   468          BVS     LISTCHK                 ;IF VS, EMPTY LIS
               50 01   D0  0127   469          MOVL    #SS$_NORMAL,R0
                        05  012A   470          RSB
                            012B   471 LISTCHK:
                  021A  30  012B   472          BSBW    EXE$EXTENDPOOL          ;ATTEMPT TO EXTEND POOL
                BC 50   E8  012E   473          BLBS    R0,EXE$ALONONPAGED      ;RETRY LISTS IF SOMETHING EXTENDED
                            0131   474 EXE$ALONPAGVAR::
               51 0F    C0  0131   475          ADDL    #MASK,R1                ;ROUND SIZE UP TO NEXT BOUNDRY
               51 0F    CA  0134   476          BICL    #MASK,R1                ;TRUNCATE SIZE BACK TO MULTIPLE
                  B1   13  0137   477          BEQL    200$                    ;IF EQL BAD ALLOCATION REQUEST
                  53   DD  0139   478          PUSHL   R3                      ;SAVE R3
          53   0000'CF     9E  013B   479          MOVAB   W^EXE$GL_NONPAGED,R3    ;GET ADDRESS OF NONPAGED MEMORY LISTHEAD
                            0140   480          DSBINT  (R3)+                   ;DISABLE INTERRUPTS
                  72   10  0146   481          BSBB    EXE$ALLOCATE            ;ALLOCATE BLOCK
```

```
                                0148   482          ENBINT                          ;ENABLE INTERRUPTS
                     53 8ED0    014B   483          POPL    R3                      ;RESTORE R3
                  01 50   E9    014E   484          BLBC    R0,EXTENDCHK            ;BR IF FAILURE
                          05    0151   485          RSB                             ;
                                0152   486 EXTENDCHK:                              ;CHECK FOR POOL EXTENSION
          0000'CF    01   C8    0152   487          BISL    #1,W^MMG$GL_NPAGNEXT    ;SET FLAG FOR EXTENSION
                01EE   30    0157   488          BSBW    EXE$EXTENDPOOL          ;ATTEMPT TO EXTEND POOL
                90 50   E8    015A   489          BLBS    R0,EXE$ALONONPAGED      ;AND REPEAT ALLOCATION ATTEMPT
          50    0124 8F    3C    015D   490          MOVZWL  #SS$_INSFMEM,R0         ;SET INSUFFICIENT MEMORY
                          05    0162   491          RSB                             ;
                                0163   492
    00000000'EF    51    D1    0163   493 SRP:      CMPL    R1,L^IOC$GL_SRPSIZE     ;CHECK FOR FIT IN SMALL PACKETS
                   C5    1A    016A   494          BGTRU   EXE$ALONPAGVAR          ;MUST USE VARIABLE POOL
                                016C   495 ;        CMPL    R1,L^IOC$GL_SRPMIN      ;CHECK FOR LOWER BOUND
                                016C   496 ;        BLSSU   EXE$ALONPAGVAR          ;MUST USE VARIABLE POOL
                   51    D5    016C   497          TSTL    R1
                   03    12    016E   498          BNEQ    10$                     ;VALID SIZE
                FF77    31    0170   499          BRW     200$                    ;INVALID SIZE
52  00000000'FF    0F    0173   500 10$:     REMQUE  @L^IOC$GL_SRPFL,R2      ;REMOVE FIRST PACEKT FROM SRP LIST
                   AF    1D    017A   501          BVS     LISTCHK                 ;IF VS, EMPTY LIST
             50    01    D0    017C   502          MOVL    #SS$_NORMAL,R0          ;
                          05    017F   503          RSB                             ;
                                0180   504
```

```
                                0180   506          .SBTTL  ALLOCATE PAGED DYNAMIC MEMORY
                                0180   507 ;+
                                0180   508 ; EXE$ALOPAGED - ALLOCATE PAGED DYNAMIC MEMORY
                                0180   509 ;
                                0180   510 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE PAGED POOL.
                                0180   511 ;
                                0180   512 ; INPUTS:
                                0180   513 ;
                                0180   514 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                                0180   515 ;
                                0180   516 ; OUTPUTS:
                                0180   517 ;
                                0180   518 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                                0180   519 ;
                                0180   520 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                                0180   521 ;
                                0180   522 ;               R1 = SIZE OF ALLOCATED BLOCK.
                                0180   523 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                                0180   524 ;-
                                0180   525
                                0180   526 EXE$ALOPAGED::                               ;ALLOCATE PAGED DYNAMIC MEMORY
          51    0F    C0      0180   527          ADDL    #MASK,R1                      ;ROUND SIZE UP TO NEXT BOUNDRY
          51    0F    CA      0183   528          BICL    #MASK,R1                      ;TRUNCATE SIZE BACK TO MULTIPLE
                28    13      0186   529          BEQL    20$                           ;IF EQL BAD ALLOCATION REQUEST
                              0188   530          SAVIPL                                ;SAVE CURRENT IPL
                54    DD      018B   531          PUSHL   R4                            ;SAVE REGISTER
     50   0000'CF    9E      018D   532          MOVAB   W^EXE$GL_PGDYNMTX,R0          ;GET ADDRESS OF PAGED MEMORY MUTEX
54   00000000'EF    D0      0192   533          MOVL    L^SCH$GL_CURPCB,R4            ;GET CURRENT PROCESS PCB ADDRESS
          FE64'    30      0199   534          BSBW    SCH$LOCKQ                     ;LOCK PAGED MEMORY DATA BASE FOR WRITE
     53   0000'CF    9E      019C   535          MOVAB   W^EXE$GL_PAGED,R3             ;GET ADDRESS OF PAGED MEMORY LISTHEAD
                17    10      01A1   536          BSBB    EXE$ALLOCATE                  ;ALLOCATE BLOCK
                07    BB      01A3   537          PUSHR   #^M<R0,R1,R2>                 ;SAVE REGISTERS
50   0000'CF    9E      01A5   538          MOVAB   W^EXE$GL_PGDYNMTX,R0          ;GET ADDRESS OF PAGED MEMORY MUTEX
          FE53'    30      01AA   539          BSBW    SCH$UNLOCK                    ;UNLOCK PAGED MEMORY DATA BASE
                17    BA      01AD   540          POPR    #^M<R0,R1,R2,R4>              ;RESTORE REGISTERS
                              01AF   541          ENBINT                                ;ENABLE INTERRUPTS
                05    01B2   542          RSB                                   ;
                              01B3   543 20$:     BUG_CHECK BADALORQSZ                  ;BAD ALLOCATION REQUEST SIZE
                50    D4      01B7   544          CLRL    R0                            ;INDICATE NO BLOCK ALLOCATED
                05    01B9   545          RSB                                   ;
                              01BA   546          .DSABL  LSB
```

```
                                01BA   548          .SBTTL  GENERAL ALLOCATE MEMORY SUBROUTINE
                                01BA   549 ;+
                                01BA   550 ; EXE$ALLOCATE - ALLOCATE MEMORY SUBROUTINE
                                01BA   551 ;
                                01BA   552 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM A POOL WHOSE ENTRIES
                                01BA   553 ; ARE MAINTAINED IN A MEMORY ORDER SORTED LIST.
                                01BA   554 ;
                                01BA   555 ; INPUTS:
                                01BA   556 ;
                                01BA   557 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                                01BA   558 ;       R3 = ADDRESS OF ALLOCATION REGION LISTHEAD.
                                01BA   559 ;
                                01BA   560 ; OUTPUTS:
                                01BA   561 ;
                                01BA   562 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                                01BA   563 ;
                                01BA   564 ;               R2 = 0 (USED TO BE THE SIZE OF LARGEST BLOCK FOUND)
                                01BA   565 ;
                                01BA   566 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                                01BA   567 ;
                                01BA   568 ;               R1 = SIZE OF ALLOCATED BLOCK.
                                01BA   569 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                                01BA   570 ;-
                                01BA   571
                                01BA   572 EXE$ALLOCATE::                               ;ALLOCATE MEMORY
     50    53    D0   01BA     573          MOVL    R3,R0                              ;COPY ADDRESS OF FIRST FREE BLOCK ADDRESS
     52    50    D0   01BD     574 10$:     MOVL    R0,R2                              ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
     50    62    D0   01C0     575          MOVL    (R2),R0                            ;GET ADDRESS OF NEXT FREE BLOCK
           1D    13   01C3     576          BEQL    30$                                ;IF EQL NO MEMORY AVAILABLE
  04 A0    51    D1   01C5     577          CMPL    R1,4(R0)                           ;FREE BLOCK BIG ENOUGH?
           F2    1A   01C9     578          BGTRU   10$                                ;IF GTRU NO
           0E    13   01CB     579          BEQL    20$                                ;IF EQL FREE BLOCK IS EXACT SIZE
53    51    50    C1   01CD     580          ADDL3   R0,R1,R3                           ;CALCULATE ADDRESS OF NEW FREE BLOCK
      83    80    D0   01D1     581          MOVL    (R0)+,(R3)+                        ;COPY LINK TO NEXT FREE BLOCK
63    60    51    C3   01D4     582          SUBL3   R1,(R0),(R3)                       ;CALCULATE SIZE OF NEW FREE BLOCK
      70    73    DE   01D8     583          MOVAL   -(R3),-(R0)                        ;SET LINK TO NEW FREE BLOCK
      62    60    D0   01DB     584 20$:     MOVL    (R0),(R2)                          ;COPY LINK TO NEW FREE BLOCK
      52    80    9E   01DE     585          MOVAB   (R0)+,R2                           ;SET ADR OF ALLOCATED BLOCK, INDICATE SUCCES
            05    01E1     586          RSB                                        ;RETURN
                  01E2     587 ;
                  01E2     588 ; NO BLOCK OF THE REQUIRED SIZE COULD BE FOUND.  RETURN 0 IN R2 WHERE THE LENGTH OF
                  01E2     589 ; THE LARGEST FREE BLOCK USED TO GO.
                  01E2     590 ;
           52     D4   01E2     591 30$:     CLRL    R2                                 ;INITIAL VALUE OF LARGEST FREE BLOCK SEEN
                  05   01E4     592          RSB                                        ;RETURN FAILURE
```

```
                     01E5   594          .SBTTL  DEALLOCATE NONPAGED DYNAMIC MEMORY
                     01E5   595 ;+
                     01E5   596 ; EXE$DEANONPAGED - DEALLOCATE NONPAGED DYNAMIC MEMORY
                     01E5   597 ;
                     01E5   598 ; THIS ROUTINE IS CALLED TO DEALLOCATE A BLOCK OF MEMORY TO A NONPAGED POOL.
                     01E5   599 ; IF THE BLOCK IS A SHARED MEMORY BLOCK TYPE, THE BLOCK IS DEALLOCATED TO
                     01E5   600 ; THE SHARED MEMORY POOL.  OTHERWISE, THE BLOCK'S ADDRESS IS CHECKED TO SEE
                     01E5   601 ; IF IT WAS ALLOCATED FROM THE I/O PACKET LOOKASIDE LIST AND IF SO,
                     01E5   602 ; IT IS RETURNED TO THAT LIST.  OTHERWISE IT IS MERGED INTO THE NORMAL
                     01E5   603 ; NONPAGED POOL.
                     01E5   604 ;
                     01E5   605 ; INPUT:
                     01E5   606 ;
                     01E5   607 ;       IF ENTRY IS AT EXE$DEANONPAGED
                     01E5   608 ;
                     01E5   609 ;               R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                     01E5   610 ;               IRP$W_SIZE(R0) = SIZE OF BLOCK TO BE DEALLOCATED.
                     01E5   611 ;               IRP$B_TYPE(R0) = TYPE OF BLOCK TO BE DEALLOCATED.
                     01E5   612 ;
                     01E5   613 ;       IF ENTRY IS AT EXE$DEANONPAGED_SIZE
                     01E5   614 ;
                     01E5   615 ;               R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                     01E5   616 ;               R1 = SIZE OF BLOCK TO BE DEALLOCATED (ASSUMED NONZERO)
                     01E5   617 ;
                     01E5   618 ; OUTPUTS:
                     01E5   619 ;
                     01E5   620 ;       THE SPECIFIED BLOCK IS RETURNED TO THE APPROPRIATE POOL.
                     01E5   621 ;-
                     01E5   622
                     01E5   623          .ENABL  LSB
                     01E5   624 EXE$DEANONPAGED::                        ;DEALLOCATE NONPAGED DYNAMIC MEMORY
          0A A0      95   01E5   625          TSTB    IRP$B_TYPE(R0)     ;IS BLOCK A SHARED MEMORY BLOCK?
             03      18   01E8   626          BGEQ    5$                 ;IF GEQ NO
           011C      31   01EA   627          BRW     EXE$DEASHARED      ;ELSE, DEALLOCATE IT
                     01ED   628
        51   08 A0   3C   01ED   629 5$:      MOVZWL  IRP$W_SIZE(R0),R1  ;GET SIZE OF BLOCK IN BYTES
                     01F1   630
                     01F1   631 EXE$DEANONPGDSIZ::                       ;DEALLOCATE NONPAGED DYNAMIC MEMORY
              50     0F   D3   01F1   632          BITL    #MASK,R0       ;CHECK PACKET ALIGNMENT
                     37   12   01F4   633          BNEQ    15$            ;BRANCH ON ERROR -- LET STD. RTN HANDLE IT
    00000000'EF      50   D1   01F6   634          CMPL    R0,L^IOC$GL_SRPSPLIT ;SMALL REQUEST PACKET?
                     0A   1F   01FD   635          BLSSU   8$             ;BR IF NOT
    00000000'FF      60   0E   01FF   636          INSQUE  (R0),@L^IOC$GL_SRPBL ;INSERT NEW PACKET AT END OF LIST
                     36   13   0206   637          BEQL    20$            ;BRANCH IF LIST WAS EMPTY, DECLARE RES. AVL.
                     05   0208   638          RSB                        ;RSB
                          0209   639
        0000'CF      50   D1   0209   640 8$:      CMPL    R0,W^EXE$GL_SPLITADR ;I/O REQUEST PACKET
                     0A   1F   020E   641          BLSSU   10$            ;IF LSSU NO
    00000000'FF      60   0E   0210   642          INSQUE  (R0),@L^IOC$GL_IRPBL ;INSERT NEW PACKET AT END OF LIST
                     25   13   0217   643          BEQL    20$            ;BRANCH IF LIST WAS EMPTY, DECLARE RES. AVL.
                     05   0219   644          RSB                        ;RETURN
                          021A   645
    00000000'EF      50   D1   021A   646 10$:     CMPL    R0,L^IOC$GL_LRPSPLIT ;LARGE REQUEST PACKET
                     0A   1F   0221   647          BLSSU   15$            ;IF LSSU, NO
    00000000'FF      60   0E   0223   648          INSQUE  (R0),@L^IOC$GL_LRPBL ;INSERT NEW PACKET AT END OF LIST
                     12   13   022A   649          BEQL    20$            ;BRANCH IF LIST WAS EMPTY, DECLARE RES. AVL.
                     05   022C   650          RSB                        ;RETURN
```

```
                                022D    651
                          53    10    022D    652 15$:    BSBB    CHECKBLOCK                  ;CHECK DEALLOCATION PARAMETERS
                    53    00^0'CF     9E    022F    653         MOVAB   W^EXE$GL_NONPAGED,R3        ;GET ADDRESS OF NONPAGED MEMORY LISTHEAD
                                0234    654         DSBINT  (R3)+                       ;DISABLE INTERRUPTS
                          5A    10    023A    655         BSBB    EXE$DEALLOCATE              ;DEALLOCATE BLOCK
                          06    11    023C    656         BRB     25$                         ;BRANCH TO DFCLARE FREE RESOURCE
                                023E    657
                                023E    658 20$:    DSBINT  #IPL$_SYNCH                 ;NEED TO ENTER SCH$RAVAIL AT IPL$_SYNC
                    50    03    3C    0244    659 25$:    MOVZWL  #RSN$_NPDYNMEM,R0           ;SET NONPAGED DYNAMIC MEMORY RESOURCE NUMBER
                          32    11    0247    660         BRB     30$                         ;
```

```
                                0249   662          .SBTTL  DEALLOCATE PAGED DYNAMIC MEMORY
                                0249   663 ;+
                                0249   664 ; EXE$DEAPAGED - DEALLOCATE PAGED DYNAMIC MEMORY
                                0249   665 ;
                                0249   666 ; THIS ROUTINE IS CALLED TO DEALLOCATE A BLOCK OF MEMORY TO THE PAGED POOL.
                                0249   667 ;
                                0249   668 ; INPUTS:
                                0249   669 ;
                                0249   670 ;       IF ENTRY IS AT EXE$DEAPAGED
                                0249   671 ;
                                0249   672 ;               R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                                0249   673 ;               IRP$W_SIZE(R0) = SIZE OF BLOCK TO BE DEALLOCATED.
                                0249   674 ;
                                0249   675 ;       IF ENTRY IS AT EXE$DEAPGDSIZ
                                0249   676 ;
                                0249   677 ;               R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                                0249   678 ;               R1 = SIZE OF BLOCK TO BE DEALLOCATED (ASSUMED NONZERO).
                                0249   679 ;
                                0249   680 ; OUTPUTS:
                                0249   681 ;
                                0249   682 ;       THE SPECIFIED BLOCK OF MEMORY IS RETURNED TO THE PAGED POOL.
                                0249   683 ;-
                                0249   684
                                0249   685 EXE$DEAPAGED::                          ;DEALLOCATE PAGED DYNAMIC MEMORY
      51    08 A0    3C         0249   686          MOVZWL  IRP$W_SIZE(R0),R1       ;GET SIZE OF BLOCK IN BYTES
                                024D   687
                                024D   688 EXE$DEAPGDSIZ::
                          33    10  024D   689          BSBB    CHECKBLOCK              ;CHECK DEALLOCATION PARAMETERS
                                024F   690          SAVIPL                          ;SAVE CURRENT IPL
                          11    BB  0252   691          PUSHR   #^M<R0,R4>              ;SAVE REGISTERS
      50    0000'CF     9E         0254   692          MOVAB   W^EXE$GL_PGDYNMTX,R0    ;GET ADDRESS OF PAGED MEMORY MUTEX
54    00000000'EF       D0         0259   693          MOVL    L^SCH$GL_CURPCB,R4      ;GET CURRENT PROCESS PCB ADDRESS
                FD9D'   30         0260   694          BSBW    SCH$LOCKW               ;LOCK PAGED MEMORY DATA BASE FOR WRITE
                   50 8ED0         0263   695          POPL    R0                      ;RESTORE REGISTER
      53    0000'CF     9E         0266   696          MOVAB   W^EXE$GL_PAGED,R3       ;GET ADDRESS OF PAGED MEMORY LISTHEAD
                   29    10  0268   697          BSBB    EXE$DEALLOCATE          ;DEALLOCATE BLOCK
      50    0000'CF     9E         026D   698          MOVAB   W^EXE$GL_PGDYNMTX,R0    ;GET ADDRESS OF PAGED MEMORY MUTEX
                FD8B'   30         0272   699          BSBW    SCH$UNLOCK              ;UNLOCK PAGED MEMORY DATA BASE
                   54 8ED0         0275   700          POPL    R4                      ;RESTORE REGISTER
            50    05    3C         0278   701          MOVZWL  #RSN$_PGDYNMEM,R0       ;SET PAGED DYNAMIC MEMORY RESOURCE NUMBER
                FD82'   30         027B   702 30$:     BSBW    SCH$RAVAIL              ;MARK RESOURCE AVAILABLE
                                027E   703          ENBINT                          ;ENABLE INTERRUPTS
                          05        0281   704          RSB                             ;
                                0282   705          .DSABL  LSB
```

```
                     0282   707          .SBTTL  CHECK BLOCK PARAMETERS SUBROUTINE
                     0282   708 ;
                     0282   709 ; CHECKBLOCK - CHECK BLOCK PARAMETERS SUBROUTINE
                     0282   710 ;
                     0282   711 ; INPUT PARAMETERS:
                     0282   712 ;
                     0282   713 ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED
                     0282   714 ;       R1 = SIZE OF BLOCK TO BE DEALLOCATED
                     0282   715 ;
                     0282   716 ; OUTPUT PARAMETER:
                     0282   717 ;
                     0282   718 ;       R1 = MODIFIED SIZE OF BLOCK TO BE DEALLOCATED
                     0282   719 ;            (SIZE IS ROUNDED UP TO NEXT MULTIPLE OF QUANTUM OF ALLOCATION)
                     0282   720 ;
                     0282   721
                     0282   722 CHECKBLOCK:                                ;CHECK BLOCK PARAMETERS
50   0F   D3   0282   723          BITL    #MASK,R0                        ;BLOCK ALIGNED ON BOUNDRY?
     08   12   0285   724          BNEQ    10$                             ;IF NEQ NO - BAD DEALLOCATION
51   0F   C0   0287   725          ADDL    #MASK,R1                        ;ROUND SIZE UP TO NEXT BOUNDRY
51   0F   CA   028A   726          BICL    #MASK,R1                        ;TRUNCATE SIZE BACK TO MULTIPLE
     06   12   028D   727          BNEQ    20$                             ;IF NEQ OKAY
                028F   728 10$:     BUG_CHECK BADDALRQSZ                    ;BAD DEALLOCATION REQUEST SIZE OR ADDRESS
     8E   D5   0293   729          TSTL    (SP)+                           ;REMOVE RETURN FROM STACK
          05   0295   730 20$:     RSB                                     ;
```

```
                     0296   732          .SBTTL  GENERAL DEALLOCATION SUBROUTINE
                     0296   733 ;+
                     0296   734 ; EXE$DEALLOCATE - DEALLOCATION SUBROUTINE
                     0296   735 ;
                     0296   736 ; INPUTS:
                     0296   737 ;
                     0296   738 ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                     0296   739 ;       R1 = SIZE OF BLOCK IN BYTES
                     0296   740 ;       R3 = ADDRESS OF ALLOCATION REGION LISTHEAD.
                     0296   741 ;
                     0296   742 ; OUTPUTS:
                     0296   743 ;
                     0296   744 ;       NONE
                     0296   745 ;-
                     0296   746
                     0296   747 EXE$DEALLOCATE::                     ;DEALLOCATE BLOCK
          54    DD   0296   748          PUSHL   R4                  ;SAVE REGISTERS
          53    DD   0298   749          PUSHL   R3
     52   53    D0   029A   750 10$:     MOVL    R3,R2               ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
     53   62    D0   029D   751          MOVL    (R2),R3             ;GET ADDRESS OF NEXT FREE BLOCK
          07    13   02A0   75           BEQL    20$                 ;IF EQL END OF LIST
     53   50    D1   02A2   7            CMPL    R0,R3               ;BLOCK LOGICALLY GO HERE?
          F3    1A   02A5   754          BGTRU   10$                 ;IF GTRU NO
          3F    13   02A7   755          BEQLU   50$                 ;IF EQLU DOUBLE DEALLOCATION
     60   53    D0   02A9   756 20$:     MOVL    R3,(R0)             ;ASSUME NO AGGLOMERATION
          11    13   02AC   757          BEQL    30$                 ;END OF LIST - NO AGGLOMERATION
54   51   50    C1   02AE   758          ADDL3   R0,R1,R4            ;CALCULATE ADDRESS OF END OF BLOCK
     54   53    D1   02B2   759          CMPL    R3,R4               ;END OF BLOCK EQUAL TO NEXT IN LIST?
          08    1A   02B5   760          BGTRU   30$                 ;IF GTR DO NOT AGGLOMERATE
          2F    1F   02B7   761          BLSSU   50$                 ;IF LSS OVERLAPPING DEALLOCATE
     60   83    D0   02B9   762          MOVL    (R3)+,(R0)          ;MOVE LINK TO BLOCK BEING RELEASED
     51   63    C0   02BC   763          ADDL    (R3),R1             ;ACCUMULATE LENGTH OF NEW FREE BLOCK
     54   52    D0   02BF   764 30$:     MOVL    R2,R4               ;CALCULATE ENDING ADDRESS OF PREVIOUS BLOCK
     82   50    D0   02C2   765          MOVL    R0,(R2)+            ;ASSUME NO AGGLOMERATION
     54   62    C0   02C5   766          ADDL    (R2),R4             ;ADD LENGTH TO BLOCK BASE ADDRESS
     54   50    D1   02C8   767          CMPL    R0,R4               ;END ADDRESS EQUAL TO BLOCK BEING RELEASED?
          0B    1A   02CB   768          BGTRU   40$                 ;IF GTR DO NOT AGGLOMERATE
          11    1F   02CD   769          BLSSU   45$                 ;IF LSS MAY BE OVERLAPPING DEALLOCATE
     51   62    C0   02CF   770          ADDL    (R2),R1             ;ACCUMULATE SIZE OF NEW FREE BLOCK
     72   60    D0   02D2   771          MOVL    (R0),-(R2)          ;MOVE LINK TO PREVIOUS FREE BLOCK
     50   52    D0   02D5   772          MOVL    R2,R0               ;SET ADDRESS OF NEW FREE BLOCK
04   A0   51    D0   02D8   773 40$:     MOVL    R1,4(R0)            ;SET SIZE OF FREE BLOCK
     53   8E    7D   02DC   774          MOVQ    (SP)+,R3            ;RESTORE REGISTERS
               05   02DF   775          RSB
                    02E0   776
                    02E0   777 45$:     ; IF WE COME HERE IT IS EITHER AN OVERLAPPING DEALLOCATE OR R2
                    02E0   778          ; (THE PREVIOUS BLOCK POINTER) IS POINTING TO THE LIST HEAD.
                    02E0   779
     52   04    C2   02E0   780          SUBL    #4,R2               ;BACK UP R2
     6E   52    D1   02E3   781          CMPL    R2,(SP)             ;IS IT POINTING TO THE LIST HEAD?
          F0    13   02E6   782          BEQL    40$                 ;YES, RESUME IN NORMAL PATH
                    02E8   783
                    02E8   784 50$:     BUG_CHECK DOUBLDEALO,FATAL      ;DOUBLE DEALLOCATION OF MEMORY BLOCK
                    02EC   785
```

```
                                02EC    787          .SBTTL  ALLOCATE A BLOCK OF SHARED MEMORY POOL
                                02EC    788 ;+
                                02EC    789 ; EXE$ALOSHARED - ALLOCATE A BLOCK OF SHARED MEMORY POOL
                                02EC    790 ;
                                02EC    791 ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE
                                02EC    792 ; SHARED MEMORY POOL.
                                02EC    793 ;
                                02EC    794 ; INPUTS:
                                02EC    795 ;
                                02EC    796 ;       R2 = ADDRESS OF SHARED MEMORY CONTROL BLOCK (SHB).
                                02EC    797 ;
                                02EC    798 ; OUTPUTS:
                                02EC    799 ;
                                02EC    800 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                                02EC    801 ;
                                02EC    802 ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                                02EC    803 ;
                                02EC    804 ;               R1 = SIZE OF ALLOCATED BLOCK.
                                02EC    805 ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                                02EC    806 ;-
                                02EC    807
                                02EC    808 EXE$ALOSHARED::                             ;ALLOCATE SHARED MEMORY POOL
     50    04 A2    D0         02EC    809          MOVL    SHB$L_DATAPAGE(R2),R0      ;GET ADDRESS OF DATA PAGE
                                02F0    810                                             ;(PAGE ALIGNED SO LOW BIT CLEAR)
           51    D4              02F0    811          CLRL    R1                         ;INIT RETRY COUNT
                                02F2    812
     52    00F8 C0    5E         02F2    813 10$:     REMQHI  SHD$Q_POOL(R0),R2          ;REMOVE A BLOCK FROM POOL
           09    1F              02F7    814          BCS     30$                        ;BR IF QUEUE LOCKED - RETRY
           06    1D              02F9    815          BVS     20$                        ;BR IF NO ENTRY - FAILURE
     51    08 A2    3C         02FB    816          MOVZWL  IRP$W_SIZE(R2),R1          ;GET SIZE OF BLOCK
           50    D6              02FF    817          INCL    R0                         ;SET SUCCESSFUL COMPLETION
           05    0301    818 20$:     RSB                                             ;
                         0302    819
EA 51      0000'CF    F3         0302    820 30$:     AOBLEQ  W^EXE$GL_LOCKRTRY,R1,10$ ;INCREMENT RETRY COUNT AND TRY AGAIN
           05    0308    821          RSB                                             ;IF RETRIES EXHAUSTED, QUEUE HEADER BAD
```

```
                     0309   823          .SBTTL  DEALLOCATE A BLOCK OF SHARED MEMORY POOL
                     0309   824 ;+
                     0309   825 ; EXE$DEASHARED - DEALLOCATE A BLOCK OF SHARED MEMORY POOL
                     0309   826 ;
                     0309   827 ; THIS ROUTINE IS CALLED TO DEALLOCATE A BLOCK OF MEMORY TO THE
                     0309   828 ; APPROPRIATE SHARED MEMORY POOL.
                     0309   829 ;
                     0309   830 ; INPUTS:
                     0309   831 ;
                     0309   832 ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                     0309   833 ;       IRP$W_SIZE(R0) = SIZE OF BLOCK TO BE DEALLOCATED.
                     0309   834 ;
                     0309   835 ; OUTPUTS:
                     0309   836 ;
                     0309   837 ;       THE SPECIFIED BLOCK IS RETURNED TO APPROPRIATE SHARED MEMORY POOL.
                     0309   838 ;
                     0309   839 ;       R0-R3 NOT PRESERVED.
                     0309   840 ;-
                     0309   841
                     0309   842 EXE$DEASHARED::                          ;DEALLOCATE SHARED MEMORY
  51    08 A0    3C  0309   843          MOVZWL  IRP$W_SIZE(R0),R1       ;GET SIZE OF BLOCK IN BYTES
        FF72     30  030D   844          BSBW    CHECKBLOCK              ;CHECK DEALLOCATION PARAMETERS
51    0000'CF    DE  0310   845          MOVAL   W^EXE$GL_SHBLIST,R1     ;GET ADDR OF SHARED MEMORY
                     0315   846                                          ; CONTROL BLOCK LISTHEAD
                     0315   847          ASSUME  SHB$L_LINK EQ 0         ;
                     0315   848 10$:
    51    61     D0  0315   849          MOVL    SHB$L_LINK(R1),R1       ;GET ADDR OF NEXT SHB
          29     13  0318   850          BEQL    40$                     ;BR IF NONE - ERROR
 04 A1    50     D1  031A   851          CMPL    R0,SHB$L_DATAPAGE(R1)   ;CHECK IF BLOCK IS FROM MEMORY
          F5     1B  031E   852          BLEQU   10$                     ;BR IF NOT
 18 A1    50     D1  0320   853          CMPL    R0,SHB$L_POOLEND(R1)    ;CHECK IF BLOCK IS FROM MEMORY
          EF     1A  0324   854          BGTRU   10$                     ;BR IF NOT
    53 04 A1    D0  0326   855          MOVL    SHB$L_DATAPAGE(R1),R3   ;GET ADDR OF DATAPAGE
          52     D4  032A   856          CLRL    R2                      ;INIT RETRY COUNT
                     032C   857 20$:
00F8 C3    60    5D  032C   858          INSQTI  (R0),SHD$Q_POOL(R3)     ;DEALLOCATE BLOCK TO POOL
          0A     1E  0331   859          BCC     30$                     ;BR IF QUEUE NOT LOCKED - SUCCESS
F3 52  0000'CF    F3  0333   860          AOBLEQ  W^EXE$GL_LOCKRTRY,R2,20$ ;INCREMENT RETRY COUNT AND TRY AGAIN
                     0339   861          BUG_CHECK BADQHDR                ;IF RETRIES EXHAUSTED, QUEUE HEADER BAD
       50    03     3C  033D   862 30$:    MOVZWL  #RSN$_NPDYNMEM,R0       ;SET NONPAGED DYNAMIC MEMORY RESOURCE NUMBER
          FCBD'     31  0340   863          BRW     MA$RAVAIL               ;REPORT RESOURCE AVAILABLE AND RETURN
                     0343   864
                     0343   865 40$:    BUG_CHECK BADDALRQSZ             ;BAD DEALLOCATION REQUEST ADDRESS
                05   0347   866          RSB                             ;
```

```
                                H 11
                     0348   868          .SBTTL  EXE$EXTENDPOOL - EXTEND NONPAGED POOL IF POSSIBLE
                     0348   869 ;+
                     0348   870 ; EXE$EXTENDPOOL - EXTEND NONPAGED POOL IF POSSIBLE
                     0348   871 ;
                     0348   872 ; THIS ROUTINE IS CALLED UPON A FAILURE TO ALLOCATE NON-PAGED POOL AND
                     0348   873 ; IT WILL ALLOCATE ADDITIONAL PAGES FOR ANY OF THE THREE SUBDIVISIONS
                     0348   874 ; OF NONPAGED POOL, THE IRP LIST, THE LRP LIST OR THE VARIABLE ALLOCATION
                     0348   875 ; AREA.
                     0348   876 ;
                     0348   877 ; INPUTS:
                     0348   878 ;
                     0348   879 ;       NONE
                     0348   880 ;
                     0348   881 ; OUTPUTS:
                     0348   882 ;
                     0348   883 ;       R0 - COMPLETION STATUS
                     0348   884 ;
                     0348   885 ;               LOW BIT CLEAR IF NO EXTENSION PERFORMED
                     0348   886 ;-
                     0348   887 EXE$EXTENDPOOL::
00000000'GF    1F    91   0348   888          CMPB    #31,G^EXE$GL_NONPAGED   ; IS INIT RUNNING?
                2F    13   034F   889          BEQL    30$                     ; YES
                50    DC   0351   890          MOVPSL  R0                      ; GET CURRENT PSL
          50    05    10   0353   891          CMPZV   #PSL$V_IPL, #PSL$S_IPL, - ; CHECK FOR IPL LESS THAN
                08         0357   892                  R0, #IPL$_SYNCH         ; OR EQUAL TO IPL$_SYNCH
                26    1B   0358   893          BLEQU   30$                     ; CONTINUE IF IPL LOW ENOUGH
            22  50    1A   E1   035A   894          BBC     #PSL$V_IS, R0, 30$      ; ALSO CONT. IF NOT ON INTERRUPT STACK
                3E    BB   035E   895          PUSHR   #^M<R1,R2,R3,R4,R5>     ; SAVE REGISTERS DESTROYED BY FORK
09 00000000'EF  00    E2   0360   896          BBSS    #0,L^IOC$GL_PFKBINT,10$ ; BR IF FORK BLOCK IN USE
   55   00000000'EF  9E   0368   897          MOVAB   L^IOC$GL_POOLFKB,R5     ; GET ADDRESS OF FORK BLOCK
                05    10   036F   898          BSBB    20$                     ; PUSH ADDRESS OF CALLER'S CALLER
                3E    BA   0371   899 10$:     POPR    #^M<R1,R2,R3,R4,R5>     ; RESTORE REGISTERS
                50    D4   0373   900          CLRL    R0                      ; INDICATE FAILURE
                05    0375   901          RSB                             ; EXIT IF ON INTERRUPT STACK
             FC87'    30   0376   902 20$:     BSBW    EXE$FORK                ; FORK TO IPL$_QUEUEAST
                           0379   903 ;
                           0379   904 ;       CONTINUATION IS AT IPL=IPL$_QUEUEAST (6) TO PERMIT SAFE ALLOCATION
                           0379   905 ;       OF PAGES FROM THE FREE PAGE LIST.
                           0379   906 ;
00000000'EF    01    CA   0379   907          BICL    #1,L^IOC$GL_PFKBINT     ; INDICATE FORK BLOCK FREE
          03FE 8F    BB   0380   908 30$:     PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9> ; SAVE REGISTERS
                           0384   909          DSBINT  L^EXE$GL_NONPAGED       ; SYNCHRONIZE DATABASE
                55    D4   038E   910          CLRL    R5                      ; ASSUME FAILURE
                           0390   911 CHECKIRP:                               ;
    52    00D0 8F    3C   0390   912          MOVZWL  #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R2 ; SIZE OF IRP
    53    0000'CF    9E   0395   913          MOVAB   W^MMG$GL_IRPNEXT,R3     ; SET ADDRESS OF NEXT VA
56   00000000'EF    9E   039A   914          MOVAB   L^IOC$GL_IRPREM,R6      ; ADDRESS OF PARTIAL PACKET
57   00000000'EF    9E   03A1   915          MOVAB   L^IOC$GL_IRPCNT,R7      ; ADDRESS OF PACKET COUNT
58   00000000'EF    9E   03A8   916          MOVAB   L^IOC$GL_IRPFL,R8       ; ADDRESS OF PACKET LIST
    59    0000'CF    9E   03AF   917          MOVAB   W^SGN$GL_IRPCNTV,R9     ; ADDRESS OF MAXIMUM COUNT
              00A5    30   03B4   918          BSBW    EXTENDLIST              ; EXTEND LIST
          55    50    C8   03B7   919          BISL    R0,R5                   ; LOGICAL OR COMPLETION STATUS
                           03BA   920 CHECKSRP:                               ;
52   00000000'EF    D0   03BA   921          MOVL    L^IOC$GL_SRPSIZE,R2     ; SIZE OF SRP
    53    0000'CF    9E   03C1   922          MOVAB   W^MMG$GL_SRPNEXT,R3     ; SET ADDRESS OF NEXT VA
56   00000000'EF    9E   03C6   923          MOVAB   L^IOC$GL_SRPREM,R6      ; ADDRESS OF PARTIAL PACKET
57   00000000'EF    9E   03CD   924          MOVAB   L^IOC$GL_SRPCNT,R7      ; ADDRESS OF PACKET COUNT
```

```
                58   00000000'EF   9E   03D4   925          MOVAB   L^IOC$GL_SRPFL,R8        ; ADDRESS OF PACKET LIST
                    59    0000'CF   9E   03DB   926          MOVAB   W^SGN$GL_SRPCNTV,R9      ; ADDRESS OF MAXIMUM COUNT
                              7A   10   03E0   927          BSBB    EXTENDLIST               ; EXTEND LIST
                          55   50   C8   03E2   928          BISL    R0,R5                    ; LOGICAL OR COMPLETION STATUS
                                         03E5   929 CHECKLRP:                               ;
                52   00000000'EF   D0   03E5   930          MOVL    L^IOC$GL_LRPSIZE,R2      ; SIZE OF LRP
                    53    0000'CF   9E   03EC   931          MOVAB   W^MMG$GL_LRPNEXT,R3      ; SET ADDRESS OF NEXT VA
                56   00000000'EF   9E   03F1   932          MOVAB   L^IOC$GL_LRPREM,R6       ; ADDRESS OF PARTIAL PACKET
                57   00000000'EF   9E   03F8   933          MOVAB   L^IOC$GL_LRPCNT,R7       ; ADDRESS OF PACKET COUNT
                58   00000000'EF   9E   03FF   934          MOVAB   L^IOC$GL_LRPFL,R8        ; ADDRESS OF PACKET LIST
                    59    0000'CF   9E   0406   935          MOVAB   W^SGN$GL_LRPCNTV,R9      ; ADDRESS OF MAXIMUM COUNT
                              4F   10   040B   936 10$:      BSBB    EXTENDLIST               ; EXTEND LIST
                          55   50   C8   040D   937          BISL    R0,R5                    ; LOGICAL OR COMPLETION STATUS
                            F8 50   E8   0410   938          BLBS    R0,10$                   ; TRY AGAIN FOR MULTIPAGE PACKETS
                                         0413   939 CHECKVAR:                               ;
                                         0413   940 ;
                                         0413   941 ; EXTEND VARIABLE AREA (R5 HAS LOW BIT SET IF ANY LIST WAS EXTENDED)
                                         0413   942 ;
                    53    0000'CF   9E   0413   943          MOVAB   W^MMG$GL_NPAGNEXT,R3     ; GET ADDRESS OF NEXT VA CELL
                              2D 63   E9   0418   944          BLBC    (R3),90$                 ; BR IF VARIABLE EXTENSION NOT NEEDED
            50  0000'CF    0000'CF   C1   041B   945          ADDL3   W^SGN$GL_NPAGEVIR,W^MMG$GL_NPAGEDYN,R0 ; COMPUTE UPPER BOUND
                          63   50   D1   0423   946          CMPL    R0,(R3)                  ; CHECK FOR AT LIMIT
                               20   1B   0426   947          BLEQU   90$                      ; BR IF AT OR PAST LIMIT
                             0081   30   0428   948          BSBW    EXTENDPAGE               ; TRY TO ADD ANOTHER PAGE
                               50   D5   042B   949          TSTL    R0                       ; CHECK STATUS
                               19   19   042D   950          BLSS    90$                      ; BR IF FAILURE
                          63   01   CA   042F   951          BICL    #1,(R3)                  ; CLEAR FLAG
             50   63   00000200 8F   C3   0432   952          SUBL3   #512,(R3),R0             ; COMPUTE ADDRESS OF NEW SEGMENT
                08 A0    0200 8F   3C   043A   953          MOVZWL  #512,IRP$W_SIZE(R0)      ; SET SIZE
                          55   01   D0   0440   954          MOVL    #1,R5                    ; RECORD SUCCESS STATUS
                             FD9F   30   0443   955          BSBW    EXE$DEANONPAGED          ; ADD NEW PAGE TO POOL
                               09   11   0446   956          BRB     100$                     ; AND EXIT
                          50   03   3C   0448   957 90$:      MOVZWL  #RSN$_NPDYNMEM,R0        ; SET RESOURCE TYPE
                       03 55   E9   044B   958          BLBC    R5,100$                  ; BR IF NOTHING AVAILABLE
                         FBAF'   30   044E   959          BSBW    SCH$RAVAIL               ; REPORT RESOURCE AVAILABLE
                50   55   D0   0451   960 100$:     MOVL    R5,R0                    ; SET COMPLETION STATUS
                                         0454   961          ENBINT                           ; RESTORE CALLER'S IPL
                03FE 8F      BA   0457   962          POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9> ; RESTORE REGISTERS
                               05   045B   963          RSB
                                         045C   964
```

```
                                045C   966          .SBTTL  EXTENDLIST - EXTEND SELECTED PACKET LIST
                                045C   967 ;+
                                045C   968 ; EXTENDLIST - EXTEND SELECTED PACKET LIST IF NEEDED
                                045C   969 ;
                                045C   970 ; THIS ROUTINE EXTENDS THE SELECTED LIST IF NECESSARY BY ADDING ONE
                                045C   971 ; PAGE AT A TIME
                                045C   972 ;
                                045C   973 ; INPUTS:
                                045C   974 ;
                                045C   975 ;       R2 = PACKET SIZE
                                045C   976 ;       R3 = ADDRESS OF MMG$GL_xRPNEXT, NEXT VA TO ALLOCATE
                                045C   977 ;       R6 = ADDRESS OF IOC$GL_xRPREM, ADDRESS OF PARTIAL PACKET
                                045C   978 ;       R7 = ADDRESS OF IOC$GL_xRPCNT, COUNT OF ALLOCATED PACKETS
                                045C   979 ;       R8 = ADDRESS OF IOC$GL_xRPFL, LIST HEADER
                                045C   980 ;       R9 = ADDRESS OF SGN$GL_xRPCNTV, MAXIMUM COUNT TO ALLOCATE
                                045C   981 ;
                                045C   982 ; OUTPUTS:
                                045C   983 ;
                                045C   984 ;       R0 = COMPLETION STATUS, LOW BIT CLEAR IF NO EXTENSION OR NOT NEEDED
                                045C   985 ;
                                045C   986 ;-
                                045C   987 EXTENDLIST:                                ;
            68    58    D1      045C   988          CMPL    R8,(R8)                   ; IS LIST EMPTY?
                  48    12      045F   989          BNEQ    70$                       ; IF NOT, EXIT
            67    69    D1      0461   990          CMPL    (R9),(R7)                 ; CHECK FOR POSSIBLE EXTENSION
                  43    15      0464   991          BLEQ    70$                       ; NO, LIST IS AT MAXIMUM SIZE
                  44    10      0466   992          BSBB    EXTENDPAGE                ; ATTEMPT TO ADD A PAGE
                  50    D5      0468   993          TSTL    R0                        ; CHECK FOR SUCCESS
                  3D    19      046A   994          BLSS    70$                       ; BR IF FAILED TO EXTEND
            50    66    D0      046C   995          MOVL    (R6),R0                   ; GET ADDRESS OF PARTIAL PACKET
                  0A    12      046F   996          BNEQ    10$                       ; BR IF ONE PRESENT
      50    63    00000200 8F   C3      0471   997          SUBL3   #512,(R3),R0      ; COMPUTE ADDRESS OF NEW PAGE
                  60    D4      0479   998          CLRL    (R0)                      ; INIT SIZE
                  51    60    D0      047B   999 10$:     MOVL    (R0),R1           ; GET SIZE OF FRAGMENT
            51    0200  C1      047E  1000          MOVAB   512(R1),R1                ; AND AUGMENT BY NEW PAGE
                  51    52    C2      0483  1001 20$:     SUBL    R2,R1             ; DIMINISH SIZE BY ONE PACKET
                  0E    19      0486  1002          BLSS    30$                       ; BRANCH IF INSUFFICIENT SPACE LEFT
               08 A0    52    B0      0488  1003          MOVW    R2,IRP$W_SIZE(R0) ; SET SIZE
                  67    D6      048C  1004          INCL    (R7)                      ; COUNT ANOTHER PACKET
                  68    60    0E      048E  1005          INSQUE  (R0),(R8)         ; INSERT ON LIST
                  50    52    C0      0491  1006          ADDL    R2,R0             ; ADVANCE ADDRESS
                  ED    11      0494  1007          BRB     20$                       ;
                  51    52    C0      0496  1008 30$:     ADDL    R2,R1             ; CORRECT SIZE OF FRAGMENT
                  05    13      0499  1009          BEQL    40$                       ; BR IF NO FRAGMENT
                  60    51    D0      049B  1010          MOVL    R1,(R0)           ; SAVE SIZE OF FRAGMENT
                  02    11      049E  1011          BRB     50$                       ;
                  50    D4      04A0  1012 40$:     CLRL    R0                        ; ZAP FRAGMENT POINTER
                  66    50    D0      04A2  1013 50$:     MOVL    R0,(R6)           ; SAVE NEW FRAGMENT POINTER
                  50    01    D0      04A5  1014 60$:     MOVL    #1,R0             ; SET SUCCESS
                        05    05      04A8  1015          RSB
                        50    D4      04A9  1016 70$:     CLRL    R0                ; SET FAILURE
                        05    05      04AB  1017          RSB
                                      04AC  1018
```

```
                                04AC  1020          .SBTTL  EXTENDPAGE - EXTEND SPECIFIED AREA BY ONE PAGE
                                04AC  1021 ;+
                                04AC  1022 ; EXTENDPAGE - EXTEND SPECIFIED AREA BY ONE PAGE
                                04AC  1023 ;
                                04AC  1024 ; THIS ROUTINE EXTENDS THE SPECIFIED AREA BY ONE PAGE PROVIDED THAT THE
                                04AC  1025 ; PAGE CAN BE ALLOCATED WITHOUT REDUCING THE FLUID PAGES BELOW THE NUMBER
                                04AC  1026 ; REQUIRED TO ACCOMODATE THE CURRENT MAXIMUM WORKING SET SIZE.  THE EXTENSION
                                04AC  1027 ; MAY ALSO BE PREVENTED IF NO PAGES ARE AVAILABLE OR THE SPECIFIED AREA IS
                                04AC  1028 ; FULL.
                                04AC  1029 ;
                                04AC  1030 ; INPUTS:
                                04AC  1031 ;
                                04AC  1032 ;       R3 = ADDRESS OF CELL CONTAINING VA TO BACK WITH PHYSICAL PAGE
                                04AC  1033 ;
                                04AC  1034 ; OUTPUTS:
                                04AC  1035 ;
                                04AC  1036 ;       R0 = COMPLETION STATUS, NEGATIVE MEANS FAILURE
                                04AC  1037 ;       @R3 = ADVANCED TO POINT TO THE NEXT PAGE IF SUCCESS
                                04AC  1038 ;
                                04AC  1039 ;       R1,R2,R3 PRESERVED
                                04AC  1040 ;-
                                04AC  1041 EXTENDPAGE:
                     50   01   CE  04AC  1042          MNEGL   #1,R0                           ; ASSUME FAILURE
                                04AF  1043          DSBINT  L^EXE$GL_NONPAGED               ; SYNCHRONIZE
                          06   BB  04B9  1044          PUSHR   #^M<R1,R2>                      ; SAVE REGISTERS
                52   63   15   EF  04BB  1045          EXTZV   #VA$V_VPN,#VA$S_VPN,(R3),R2     ; GET VIRTUAL PAGE NUMBER
                   52 0000'DF42   DE  04C0  1046          MOVAL   @W^MMG$GL_SPTBASE[R2],R2        ; COMPUTE SVAPTE
                          62   D5  04C6  1047          TSTL    (R2)                            ; CHECK FOR EMPTY
                          5D   12  04C8  1048          BNEQ    90$                             ; BR IF FILLED
                     51   0000'CF   3C  04CA  1049          MOVZWL  W^MPW$GW_LOLIM,R1               ; GET LOW LIMIT FOR MODIFY LIST
                     51   0000'CF   C0  04CF  1050          ADDL    W^SGN$GL_FREELIM,R1             ; ADD FREE LIST LIMIT
51   00000000'EF   51   C3  04D4  1051          SUBL3   R1,L^PFN$GL_PHYPGCNT,R1         ; AND COMPUTE NET FLUID PAGE COUNT
                0000'CF   51   D1  04DC  1052          CMPL    R1,W^SGN$GL_MAXWSCNT            ; COMPARE WITH MAX WORKING SET
                          44   15  04E1  1053          BLEQ    90$                             ; BR IF NO ROOM FOR GROWTH
                          0E   BB  04E3  1054          PUSHR   #^M<R1,R2,R3>                   ; SAVE VOLATILE REGISTERS
                      FB18'    30  04E5  1055          BSBW    MMG$ALLOCPFN                    ; AND ATTEMPT TO ALLOCATE A PFN
                          0E   BA  04E8  1056          POPR    #^M<R1,R2,R3>                   ; RESTORE VOLATILE REGISTERS
                          50   D5  04EA  1057          TSTL    R0                              ; CHECK STATUS
                          39   19  04EC  1058          BLSS    90$                             ; BR IF NO PAGE ALLOCATED
             00000000'EF   01   8A  04EE  1059          BICB    #1,IOC$GL_POOLEXP_STS           ; POOL SUCCESSFULLY EXPANDED
                   0000'DF40   B6  04F5  1060          INCW    @W^PFN$AW_REFCNT[R0]            ; REFERENCE COUNT
               0000'DF40   52   D0  04FA  1061          MOVL    R2,@W^PFN$AL_PTE[R0]            ; PTE BACK-POINTER
               0000'DF40   07   90  0500  1062          MOVB    #PFN$C_ACTIVE,@W^PFN$AB_STATE[R0] ; STATE IS ACTIVE
               0000'DF40   01   90  0506  1063          MOVB    #1,@W^PFN$AB_TYPE[R0]           ; TYPE IS SYSTEM PAGE
62   50   B0000000 8F   C9  050C  1064          BISL3   #<PTE$C_ERKW!PTE$M_VALID>,R0,(R2) ; SET PAGE INTO SPT
             00000000'EF   D7  0514  1065          DECL    L^PFN$GL_PHYPGCNT               ; ONE LESS FLUID PAGE
     63      00000200 8F   C0  051A  1066          ADDL    #512,(R3)                       ; ADVANCE POINTER TO NEXT PAGE
                          06   BA  0521  1067 80$:       POPR    #^M<R1,R2>                      ; RESTORE REGISTERS
                               0523  1068          ENBINT                                  ; RESTORE IPL
                               05  0526  1069          RSB
                                0527  1070 ;
                                0527  1071 ; Pool expansion failure
                                0527  1072 ;
  F2 00000000'EF   00   E2  0527  1073 90$:       BBSS    #0,IOC$GL_POOLEXP_STS,80$       ; MESSAGE ALREADY OUTPUT
  EA 00000000'EF   00   E5  052F  1074          BBCC    #0,IOC$GT_NOPOOL_TWP,80$        ; TWP IN USE
     52   00000030'EF   9E  0537  1075          MOVAB   IOC$GT_NOPOOL_TWP+TTY$K_WB_LENGTH,R2 ; MESSAGE ADDRESS
     51   00000002'EF   3C  053E  1076          MOVZWL  IOC$GL_POOLEXP_STS+2,R1         ; MESSAGE LENGTH
```

```
                00000000'EF   16  0545  1077          JSB     IOC$CONBRDCST                   ; OUTPUT THE MESSAGE
                     50   01   CE  054B  1078          MNEGL   #1,R0                           ; INDICATE EXPANSION FAILURE
                          D1   11  054E  1079          BRB     80$
                                   0550  1080
```

```
                     0550  1082          .SBTTL  ALLOCATE PHYSICALLY-CONTIGUOUS MEMORY
                     0550  1083 ;++
                     0550  1084 ; EXE$ALOPHYCNTG
                     0550  1085 ;
                     0550  1086 ; Allocate and map to system virtual address space N physically-contiguous
                     0550  1087 ; pages.
                     0550  1088 ;
                     0550  1089 ; Inputs:
                     0550  1090 ;       R1 = N = number of physically-contiguous pages
                     0550  1091 ;
                     0550  1092 ; Implicit Inputs:
                     0550  1093 ;       IPL = SYNCH
                     0550  1094 ;
                     0550  1095 ; Outputs:
                     0550  1096 ;       R0 = status: SUCCESS, INSFMEM, INSFSPTS
                     0550  1097 ;       R1 = preserved
                     0550  1098 ;       R2 = system virtual address of N pages of memory if success
                     0550  1099 ;       all other registers preserved
                     0550  1100 ;
                     0550  1101 ; Implicit Outputs:
                     0550  1102 ;       None.
                     0550  1103 ;
                     0550  1104 ; Side Effects:
                     0550  1105 ;       MMG$ALLOCONTIG called
                     0550  1106 ;       IOC$ALLOSPT called - so SPTs are allocated
                     0550  1107 ;
                     0550  1108 ;--
                     0550  1109 EXE$ALOPHYCNTG::
            3A   BB  0550  1110          PUSHR   #^M<R1,R3,R4,R5>        ; save work registers
                     0552  1111                                          ;  r1 = input used as loop counter
                     0552  1112                                          ;  r3 = address of SPT
                     0552  1113                                          ;  r4 = index into PFN database
                     0552  1114                                          ;  r5 = temp storage
00000000'EF  16      0552  1115          JSB     MMG$ALLOCONTIG          ; find N physically-contiguous pages
                     0558  1116 ;
                     0558  1117 ; MMG$ALLOCONTIG returns:
                     0558  1118 ;       R0 = first PFN in range, or negative if cannot fulfill request
                     0558  1119 ;
      55   50    D0  0558  1120          MOVL    R0,R5                   ; success?
           56    19  055B  1121          BLSS    20$                     ; get out if none found
         FAA0'   30  055D  1122          BSBW    IOC$ALLOSPT             ; allocate N SPTs to map VAs
        57 50    E9  0560  1123          BLBC    R0,30$                  ; if LBC, no system page table slots
                     0563  1124 ;
                     0563  1125 ; IOC$ALLOSPT returns:
                     0563  1126 ;       R1 = preserved, R2 = SVPN (index into SPT), R3 = address of SPT
                     0563  1127 ;
                     0563  1128 ; The main loop indexes backwards through the system page table entries
                     0563  1129 ; and backwards through the PFN database.  It goes backwards so that the
                     0563  1130 ; last system virtual address calculated can be returned to the caller.
                     0563  1131 ;
                     0563  1132 ;                                         ;  r0 = index into SPT
50   52   51   C1     0563  1133          ADDL3   R1,R2,R0                ; start at last SPT and go backwards
54   51   55   C1     0567  1134          ADDL3   R5,R1,R4                ; start PFNs at end in loop
                     056B  1135 10$:                                     ; set up system page-table entry
           50   D7   056B  1136          DECL    R0                      ; back up SVPN index
           54   D7   056D  1137          DECL    R4                      ; back up PFN index
     6340  54   D0   056F  1138          MOVL    R4,(R3)[R0]             ; fill PFN in SPT
```

```
                 6340   90000000 8F    C8  0573  1139          BISL2   #<PTE$C_KW!PTE$M_VALID>,(R3)[R0] ; kernel mode access, valid
                                       057B  1140 ;
                                       057B  1141 ; Set up PFN database
                                       057B  1142 ;
      00000000'FF44     01    B0      057B  1143          MOVW    #1,@L^PFN$AW_REFCNT[R4]          ; reference count=1
      00000000'FF44     07    90      0583  1144          MOVB    #PFN$C_ACTIVE,@L^PFN$AB_STATE[R4] ; state=active
      00000000'FF44     01    90      058B  1145          MOVB    #PFN$C_SYSTEM,@L^PFN$AB_TYPE[R4]  ; type=system
00000000'FF44     6340    DE      0593  1146          MOVAL   (R3)[R0],@L^PFN$AL_PTE[R4]       ; PTE points to SPT
                                       059C  1147 ;
                                       059C  1148 ; Invalidate system virtual address
                                       059C  1149 ;
           52   50    09    78      059C  1150          ASHL    #9,R0,R2                         ; turn SVPN into VA
        52   80000000 8F    C8      05A0  1151          BISL2   #<1@31>,R2                       ; make VA a system VA
                                       05A7  1152          INVALID R2                       ; and clear translation buffer
                  BE 51    F5      05AA  1153          SOBGTR  R1,10$                   ; loop N times
                                       05AD  1154
              50    01    D0      05AD  1155          MOVL    #SS$_NORMAL,R0           ; indicates success (R2 has system VA)
                    3A    BA      05B0  1156 15$:     POPR    #^M<R1,R3,R4,R5>         ; restore work registers
                                       05B2  1157          RSB                              ;
           50   0124 8F    3C      05B3  1158 20$:     MOVZWL  #SS$_INSFMEM,R0          ; no memory available
                    F6    11      05B8  1159          BRB     15$                      ; return
           50   2044 8F    3C      05BA  1160 30$:     MOVZWL  #SS$_INSFSPTS,R0         ; no SPTs left
                    EF    11      05BF  1161          BRB     15$                      ; return
                                       05C1  1162
```

```
00000000  1164          .PSECT  Y$EXEPAGED
     0000  1165          .SBTTL  *** THE FOLLOWING ROUTINES ARE IN THE PAGED EXEC ***
     0000  1166          .SBTTL  EXE$ALOP1PROC - ALLOCATE MEMORY FROM PROCESS ALLOCATION REGION
     0000  1167
     0000  1168 ;+
     0000  1169 ; EXE$ALOP1PROC - ALLOCATE MEMORY FROM PROCESS ALLOCATION REGION.
     0000  1170 ;
     0000  1171 ; This routine is called to allocate memory from the Process Allocation Region
     0000  1172 ; for those requests which need memory beyond the life of an image.
     0000  1173 ;
     0000  1174 ; INPUTS:
     0000  1175 ;
     0000  1176 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
     0000  1177 ;
     0000  1178 ; OUTPUTS:
     0000  1179 ;
     0000  1180 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
     0000  1181 ;
     0000  1182 ;       R0 = LOW BIT SET IF MEMRY ALLOCATED WITH:
     0000  1183 ;
     0000  1184 ;               R1 = SIZE OF ALLOCATED BLOCK
     0000  1185 ;               R2 = ADDRESS OF ALLOCATED BLOCK
     0000  1186 ;
     0000  1187 ;       R3 Destroyed.
     0000  1188 ;
     0000  1189 ; ENVIRONMENT:
     0000  1190 ;
     0000  1191 ;       Called from EXEC or KERNEL modes, this routine will switch
     0000  1192 ;       into KERNEL mode and disable ASTs to perform the actual
     0000  1193 ;       allocation.
     0000  1194 ;
     0000  1195 ;       IPL =  0 or ASTDEL
     0000  1196 ;
     0000  1197 ;--
     0000  1198
     0000  1199 EXE$ALOP1PROC::
     0000  1200          ADDL    #MASK,R1                        ; round size to next boundry
     0003  1201          BICL    #MASK,R1                        ; truncate size back to multiple
     0006  1202          BSBW    P1SYNCH                         ; get to correct mode, AST level
     0009  1203
     0009  1204 ;
     0009  1205 ; Now in Kernel Mode with ASTs disabled
     0009  1206 ;
     0009  1207
     0009  1208          MOVAL   a#CTL$GQ_ALLOCREG,R3            ; get list head address
     0010  1209          BSBW    EXE$ALLOCATE                    ; allocate block
     0013  1210          BLBC    R0,10$                          ; get out on error
     0016  1211          SUBL    R1,a#CTL$GL_PRCALLCNT           ; keep track of space in use
     001D  1212 10$:     RSB

51 0F C0
51 0F CA
00E0 30
00000000'9F DE
01A7' 30
07 50 E9
00000000'9F 51 C2
05
```

```
                     001E  1214          .SBTTL  EXE$ALOP1IMAG - ALLOCATE MEMORY FROM PROCESS ALLOCATION REGION
                     001E  1215
                     001E  1216 ;+
                     001E  1217 ; EXE$ALOP1IMAG - ALLOCATE MEMORY FROM PROCESS ALLOCATION REGION.
                     001E  1218 ;
                     001E  1219 ; This routine is called to  allocate  memory  from  the  Process  Allocation
                     001E  1220 ; Region  for those requests which need memory only for the life of an image.
                     001E  1221 ; This routine will expand P0 space to extend the process  allocation  region
                     001E  1222 ; when:
                     001E  1223 ;
                     001E  1224 ;       1.  There is not enough space left in the Process Allocation Region to
                     001E  1225 ;           satisfy the request, or
                     001E  1226 ;
                     001E  1227 ;       2.  The space left in the  Process  Allocation  Region  is  less  than
                     001E  1228 ;           SGN$GW_CTLPAGES - SGN$GW_CTLIMGLIM, and
                     001E  1229 ;
                     001E  1230 ;       3.  The linker option NOP0BUFS has not been selected.
                     001E  1231 ;
                     001E  1232 ;       4.  The routine is called at IPL 0.
                     001E  1233 ;
                     001E  1234 ; INPUTS:
                     001E  1235 ;
                     001E  1236 ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                     001E  1237 ;
                     001E  1238 ; OUTPUTS:
                     001E  1239 ;
                     001E  1240 ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                     001E  1241 ;
                     001E  1242 ;       R0 = LOW BIT SET IF MEMRY ALLOCATED WITH:
                     001E  1243 ;
                     001E  1244 ;               R1 = SIZE OF ALLOCATED BLOCK
                     001E  1245 ;               R2 = ADDRESS OF ALLOCATED BLOCK
                     001E  1246 ;
                     001E  1247 ;       R3 Destroyed.
                     001E  1248 ;
                     001E  1249 ; ENVIRONMENT:
                     001E  1250 ;
                     001E  1251 ;       Called from EXEC or KERNEL modes, this routine will switch
                     001E  1252 ;       into KERNEL mode and disable ASTs to perform the actual
                     001E  1253 ;       allocation.
                     001E  1254 ;
                     001E  1255 ;       IPL = 0 or ASTDEL
                     001E  1256 ;
                     001E  1257 ;--
                     001E  1258
                     001E  1259 EXE$ALOP1IMAG::
     51    0F    C0  001E  1260          ADDL    #MASK,R1                        ; round size to next boundry
     51    0F    CA  0021  1261          BICL    #MASK,R1                        ; truncate size back to multiple
        00C2    30  0024  1262          BSBW    P1SYNCH                         ; get to correct mode, AST level
                    0027  1263
                    0027  1264 ;
                    0027  1265 ; Now in Kernel Mode with ASTs disabled
                    0027  1266 ;
                    0027  1267
                    0027  1268
00000000'9F         D5  0027  1269          TSTL    @#CTL$GL_PRCALLCNT              ; can we get any P1?
           24       15  002D  1270          BLEQ    POSPACE                         ; branch if not.
```

```
                53   00000000'9F   DE   002F  1271          MOVAL   a#CTL$GQ_ALLOCREG,R3                    ; get list head address
                               0181'  30   0036  1272          BSBW    EXE$ALLOCATE                            ; allocate block
                            17 50     E9   0039  1273          BLBC    R0,POSPACE                              ; on error go try P0 space
                00000000'9F    51     C2   003C  1274          SUBL    R1,a#CTL$GL_PRCALLCNT                   ; keep track of space in use
                               01     19   0043  1275          BLSS    10$                                     ; get P0 space if not enough
                                      05   0045  1276          RSB
                                           0046  1277
                00000000'9F    51     C0   0046  1278 10$:     ADDL    R1,a#CTL$GL_PRCALLCNT                   ; backup count
                         50    52     D0   004D  1279          MOVL    R2,R0                                   ; set up for DEALLOCATE
                            0243'  30   0050  1280          BSBW    EXE$DEALLOCATE                          ; return space
                                           0053  1281
                                           0053  1282 POSPACE:
                         50    D4   0053  1283          CLRL    R0                                      ; assume failure
             6A 00000000'9F    05   E0   0055  1284          BBS     #IMP$V_NOPOBUFS,a#PIO$GW_IIOIMPA,40$ ; branch if cant expand P0
                53   00000000'9F    DE   005D  1285          MOVAL   a#CTL$GQ_POALLOC,R3                     ; get P0 list head
                                           0064  1286
                            0153'  30   0064  1287 10$:     BSBW    EXE$ALLOCATE                            ; try to get space
                         5D 50     E8   0067  1288          BLBS    R0,40$                                  ; branch if available
                                           006A  1289
                                           006A  1290 ;
                                           006A  1291 ; SPACE NOT AVAILABLE ON P0 LISTHEAD, EXPAND REGION TO GET IT
                                           006A  1292 ;
                                           006A  1293
                         04 AE     D5   006A  1294          TSTL    4(SP)                                   ; can we lower IPL?
                            58     12   006D  1295          BNEQ    40$                                     ; branch if not
                                        006F  1296          SETIPL  #0                                      ; lower IPL
                            51     DD   0072  1297          PUSHL   R1                                      ; save request size
                51   51    F7 8F     78   0074  1298          ASHL    #-9,R1,R1                               ; get size of request in pages
                               51     D6   0079  1299          INCL    R1                                      ; one more page
                      10       51     D1   007B  1300          CMPL    R1,#16                                  ; at least 16 pages?
                               03     18   007E  1301          BGEQ    20$                                     ; branch if so
                      51       10     D0   0080  1302          MOVL    #16,R1                                  ; set 16 pages then
                      52       7E     7E   0083  1303 20$:     MOVAQ   -(SP),R2                                ; get space for RETADR
                                           0086  1304          $EXPREG_S -                                     ; expand P0
                                           0086  1305                    PAGCNT  =       R1, -
                                           0086  1306                    RETADR  =       (R2), -
                                           0086  1307                    ACMODE  =       #PSL$C_KERNEL, -
                                           0086  1308                    REGION  =       #0
                         2B 50     E9   0095  1309          BLBC    R0,30$                                  ; get out on error
                                           0098  1310          $SETPRT_S -                                     ; change page protection
                                           0098  1311                    INADR   =       (R2), -
                                           0098  1312                    ACMODE  =       #PSL$C_KERNEL, -
                                           0098  1313                    PROT    =       #PRT$C_UREW
                         13 50     E9   00AD  1314          BLBC    R0,30$                                  ; get out on error
                      50    8E     7D   00B0  1315          MOVQ    (SP)+,R0                                ; get address range
                      51    50     C2   00B3  1316          SUBL    R0,R1                                   ; get size of area
                            51     D6   00B6  1317          INCL    R1
                                        00B8  1318          SETIPL  #IPL$_ASTDEL                            ; restore IPL
                               01D8'  30   00BB  1319          BSBW    EXE$DEALLOCATE                          ; put new space on list
                      51       8E     D0   00BE  1320          MOVL    (SP)+,R1                                ; restore R1
                               A1     11   00C1  1321          BRB     10$                                     ; go back and try request
                                           00C3  1322
             5E    0C AE     DE   00C3  1323 30$:     MOVAL   12(SP),SP                               ; pop temp space, saved R1
                               05   00C7  1324 40$:     RSB                                             ; and return
                                           00C8  1325
```

```
                        00C8  1327          .SBTTL  EXE$DEAP1 - RETURN MEMORY TO PROCESS ALLOCATION REGION
                        00C8  1328
                        00C8  1329 ;+
                        00C8  1330 ; EXE$DEAP1 - RETURN MEMORY TO PROCESS ALLOCATION REGION
                        00C8  1331 ;
                        00C8  1332 ; This routine is called to return memory allocated via the EXE$ALOP1IMAG
                        00C8  1333 ; or EXE$ALOP1PROC routines.
                        00C8  1334 ;
                        00C8  1335 ; INPUTS:
                        00C8  1336 ;
                        00C8  1337 ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                        00C8  1338 ;       R1 = SIZE OF BLOCK IN BYTES
                        00C8  1339 ;
                        00C8  1340 ; OUTPUTS:
                        00C8  1341 ;
                        00C8  1342 ;       R0-R3 Destroyed.
                        00C8  1343 ;
                        00C8  1344 ; ENVIRONMENT:
                        00C8  1345 ;
                        00C8  1346 ;       Called from EXEC or KERNEL modes, this routine will switch
                        00C8  1347 ;       into KERNEL mode and disable ASTs to perform the actual
                        00C8  1348 ;       de-allocation.
                        00C8  1349 ;
                        00C8  1350 ;       IPL = 0 or ASTDEL
                        00C8  1351 ;-
                        00C8  1352
                        00C8  1353 EXE$DEAP1::
                        00C8  1354          ;BSBW   CHECKBLOCK                          ; check deall parameters
                  1F 10 00C8  1355          BSBB    P1SYNCH                             ; get to correct mode, AST l
                        00CA  1356
                        00CA  1357 ;
                        00CA  1358 ; Now in Kernel Mode with ASTs disabled
                        00CA  1359 ;
                        00CA  1360
                10 50 1E E1 00CA  1361          BBC     #30,R0,10$                      ; branch if P0 address
           53 00000000'9F DE 00CE  1362          MOVAL   a#CTL$GQ_ALLOCREG,R3            ; get list head address
00000000'9F 51 C0 00D5  1363          ADDL    R1,a#CTL$GL_PRCALLCNT           ; keep track of space
                   07 11 00DC  1364          BRB     20$
                        00DE  1365
           53 00000000'9F DE 00DE  1366 10$:     MOVAL   a#CTL$GQ_POALLOC,R3             ; get P0 list head
                  01AE' 30 00E5  1367 20$:     BSBW    EXE$DEALLOCATE                  ; return space
                        05 00E8  1368          RSB
                        00E9  1369
```

```
                        00E9  1371          .SBTTL  P1 SYNCH ROUTINE
                        00E9  1372
                        00E9  1373 ;+
                        00E9  1374 ;
                        00E9  1375 ; This routine is called to switch into kernel mode if necessary.  It
                        00E9  1376 ; calls the subroutine P1KERN via a $CMKRNL service or a CALLG instruction
                        00E9  1377 ; depending on the current mode.  The P1KERN subroutine performs a JSB
                        00E9  1378 ; back to P1SYNCH's return address to perform the actual allocation
                        00E9  1379 ; routine after disabling ASTs.  Upon return from P1KERN, P1SYNCH
                        00E9  1380 ; pops it's return address and RSBs to it's caller's caller.
                        00E9  1381 ;
                        00E9  1382 ;-
                        00E9  1383
                        00E9  1384
                        00E9  1385 P1SYNCH:
        52   DC          00E9  1386          MOVPSL  R2                      ; get PSL
  17 52  18  E1         00EB  1387          BBC     #PSL$V_CURMOD,R2,10$    ; branch if not in EXEC mode
        03   BB         00EF  1388          PUSHR   #^M<R0,R1>              ; push input args
    53  5E   D0         00F1  1389          MOVL    SP,R3                   ; get argument list address
                        00F4  1390          $CMKRNL_S -                     ; go into kernel mode
                        00F4  1391                  ROUTIN = 30$, -
                        00F4  1392                  ARGLST = (R3)
        0E   BA         0103  1393          POPR    #^M<R1,R2,R3>           ; get return R1,R2, pop return addr
             05         0105  1394          RSB                             ; return to caller's caller.
                        0106  1395
    53  8E   D0         0106  1396 10$:     MOVL    (SP)+,R3                ; get routine address
                        0109  1397
                        0109  1398 20$:     DSBINT  #IPL$_ASTDEL            ; raise IPL
        63   16         010F  1399          JSB     (R3)                    ; go back to routine
                        0111  1400          ENBINT                          ; lower IPL
             05         0114  1401          RSB
                        0115  1402
                  000C  0115  1403 30$:     .WORD   ^M<R2,R3>
    50  6C   7D         0117  1404          MOVQ    (AP),R0                 ; get input args
53  08  AC   D0         011A  1405          MOVL    8(AP),R3                ; get routine address
        E9   10         011E  1406          BSBB    20$                     ; synch and call routine
    6C  51   7D         0120  1407          MOVQ    R1,(AP)                 ; return R1,R2 values
             04         0123  1408          RET
                        0124  1409
                        0124  1410          .END
```

```
$$T1                    = 00000000
BUG$_BADALORQSZ           ********    X   02
BUG$_BADDALRQSZ           ********    X   02
BUG$_BADQHDR              ********    X   02
BUG$_DOUBLDEAL O          ********    X   02
CEB$C_LENGTH            = 00000038
CEB$C_SLAVLNG           = 00000044
CHECKBLOCK                00000282 R      02
CHECKIRP                  00000390 R      02
CHECKLRP                  000003E5 R      02
CHECKSRP                  000003BA R      02
CHECKVAR                  00000413 R      02
CTL$GL_PRCALLCNT          ********    X   03
CTL$GQ_ALLOCREG           ********    X   03
CTL$GQ_POALLOC            ********    X   03
DYN$C_BUFIO             = 00000013
DYN$C_CEB               = 00000004
DYN$C_IRP               = 0000000A
DYN$C_JIB               = 0000002F
DYN$C_PCB               = 0000000C
DYN$C_TQE               = 0000000F
EXE$ALLOCATE              000001BA RG     02
EXE$ALLOCBUF              00000061 RG     02
EXE$ALLOCCEB              00000000 RG     02
EXE$ALLOCIRP              00000012 RG     02
EXE$ALLOCJIB              00000009 RG     02
EXE$ALLOCPCB              0000001A RG     02
EXE$ALLOCTQE              00000023 RG     02
EXE$ALONONPAGED           000000ED RG     02
EXE$ALONPAGVAR            00000131 RG     02
EXE$ALONPAGWAIT           0000009C RG     02
EXE$ALONPAGWAITS          0000008F RG     02
EXE$ALOP1IMAG             0000001E RG     03
EXE$ALOP1PROC             00000000 RG     03
EXE$ALOPAGED              00000180 RG     02
EXE$ALOPAGWAIT            000000AF RG     02
EXE$ALOPHYCNTG            00000550 RG     02
EXE$ALOSHARED             000002EC RG     02
EXE$C_ALCGRNMSK         = 0000000F  G
EXE$C_CMSTKSZ             ********    X   02
EXE$DEALLOCATE            00000296 RG     02
EXE$DEANONPAGED           000001E5 RG     02
EXE$DEANONPGDSIZ          000001F1 RG     02
EXE$DEAP1                 000000C8 RG     03
EXE$DEAPAGED              00000249 RG     02
EXE$DEAPGDSIZ             0000024D RG     02
EXE$DEASHARED             00000309 RG     02
EXE$EXTENDPOOL            00000348 RG     02
EXE$FORK                  ********    X   02
EXE$GL_LOCKRTRY           ********    X   02
EXE$GL_NONPAGED           ********    X   02
EXE$GL_PAGED              ********    X   02
EXE$GL_PGDYNMTX           ********    X   02
EXE$GL_SHBLIST            ********    X   02
EXE$GL_SPLITADR           ********    X   02
EXTENDCHK                 00000152 R      02
EXTENDLIST                0000045C R      02
EXTENDPAGE                000004AC R      02
IMP$V_NOPOBUFS          = 00000005
IOC$ACLOSPT               ********    X   02
IOC$CONBRDCST             ********    X   02
IOC$GL_IRPBL              ********    X   02
IOC$GL_IRPCNT             ********    X   02
IOC$GL_IRPFL              ********    X   02
IOC$GL_IRPMIN             ********    X   02
IOC$GL_IRPREM             ********    X   02
IOC$GL_LRPBL              ********    X   02
IOC$GL_LRPCNT             ********    X   02
IOC$GL_LRPFL              ********    X   02
IOC$GL_LRPMIN             ********    X   02
IOC$GL_LRPREM             ********    X   02
IOC$GL_LRPSIZE            ********    X   02
IOC$GL_LRPSPLIT           ********    X   02
IOC$GL_PFKBINT            ********    X   02
IOC$GL_POOLEXP_STS        ********    X   02
IOC$GL_POOLFKB            ********    X   02
IOC$GL_SRPBL              ********    X   02
IOC$GL_SRPCNT             ********    X   02
IOC$GL_SRPFL              ********    X   02
IOC$GL_SRPREM             ********    X   02
IOC$GL_SRPSIZE            ********    X   02
IOC$GL_SRPSPLIT           ********    X   02
IOC$GT_NOPOOL_TWP         ********    X   02
IPL$_ASTDEL             = 00000002
IPL$_SYNCH              = 00000008
IRP$B_TYPE              = 0000000A
IRP$C_LENGTH            = 000000C4
IRP$W_SIZE              = 00000008
JIB$C_LENGTH            = 00000074
LISTCHK                   0000012B R      02
LRP                       0000010C R      02
MA$RAVAIL                 ********    X   02
MASK                    = 0000000F
MMG$ALLOCONTIG            ********    X   02
MMG$ALLOCPFN              ********    X   02
MMG$GL_IRPNEXT            ********    X   02
MMG$GL_LRPNEXT            ********    X   02
MMG$GL_NPAGEDYN           ********    X   02
MMG$GL_NPAGNEXT           ********    X   02
MMG$GL_SPTBASE            ********    X   02
MMG$GL_SRPNEXT            ********    X   02
MPW$GW_LOLIM              ********    X   02
POSPACE                   00000053 R      03
P1SYNCH                   000000E9 R      03
PCB$C_LENGTH            = 00000120
PCB$L_STS               = 00000024
PCB$V_SSRWAIT           = 0000000A
PFN$AB_STATE              ********    X   02
PFN$AB_TYPE               ********    X   02
PFN$AL_PTE                ********    X   02
PFN$AW_REFCNT             ********    X   02
PFN$C_ACTIVE            = 00000007
PFN$C_SYSTEM            = 00000001
PFN$GL_PHYPGCNT           ********    X   02
```

```
PIO$GW_IIOIMPA                  ********    X   03
PR$_IPC                       = 00000012
PR$_TBIS                      = 0000003A
PRT$C_UREW                      ********    X   03
PSL$C_KERNEL                  = 00000000
PSL$S_IPL                     = 00000005
PSL$V_CURMOD                  = 00000018
PSL$V_IPL                     = 00000010
PSL$V_IS                      = 0000001A
PTE$C_ERKW                    = 30000000
PTE$C_KW                      = 10000000
PTE$M_VALID                   = 80000000
RSN$_NPDYNMEM                 = 00000003
RSN$_PGDYNMEM                 = 00000005
RWAIT_CHECK_NP                  000000C1 R      02
RWAIT_CHECK_PAG                 000000BD R      02
SCH$GL_CURPCB                   ********    X   02
SCH$LOCKW                       ********    X   02
SCH$RAVAIL                      ********    X   02
SCH$RWAIT                       ********    X   02
SCH$UNLOCK                      ********    X   02
SGN$GL_FREELIM                  ********    X   02
SGN$GL_IRPCNTV                  ********    X   02
SGN$GL_LRPCNTV                  ********    X   02
SGN$GL_MAXWSCNT                 ********    X   02
SGN$GL_NPAGEVIR                 ********    X   02
SGN$GL_SRPCNTV                  ********    X   02
SHB$L_DATAPAGE                = 00000004
SHB$L_LINK                    = 00000000
SHB$L_POOLEND                 = 00000018
SHD$Q_POOL                    = 000000F8
SRP                             00000163 R      02
SS$_INSFMEM                   = 00000124
SS$_INSFSPTS                  = 00002044
SS$_NORMAL                    = 00000001
SYS$CMKRNL                      ********   GX   03
SYS$EXPREG                      ********   GX   03
SYS$SETPRT                      ********   GX   03
TQE$C_LENGTH                  = 00000030
TTY$K_WB_LENGTH               = 00000030
VA$S_VPN                      = 00000015
VA$V_VPN                      = 00000009
```

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
AEXENONPAGED	000005C1 (1473.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
Y$EXEPAGED	00000124 (292.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.04	00:00:02.34
Command processing	121	00:00:00.48	00:00:05.19
Pass 1	436	00:00:16.79	00:00:48.36
Symbol table sort	0	00:00:02.50	00:00:08.04
Pass 2	257	00:00:04.34	00:00:14.95
Symbol table output	25	00:00:00.16	00:00:00.50
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	873	00:00:24.34	00:01:19.41

The working set limit was 1800 pages.
99738 bytes (195 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1604 non-local and 67 local symbols.
1410 source lines were read in Pass 1, producing 25 object records in Pass 2.
40 pages of virtual memory were used to define 39 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	23
_255DUA28:[SYSLIB]STARLET.MLB;2	13
TOTALS (all libraries)	36

1737 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

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