L 12

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

FILEID**MEMORYALC


```
MM      MM  EEEEEEEEEE  MM      MM    OOOOOO    RRRRRRRR    YY      YY    AAAAAA    LL            CCCCCCCC
MM      MM  EEEEEEEEEE  MM      MM    OOOOOO    RRRRRRRR    YY      YY    AAAAAA    LL            CCCCCCCC
MMMM  MMMM  EE          MMMM  MMMM  OO      OO  RR      RR  YY      YY  AA      AA  LL          CC
MMMM  MMMM  EE          MMMM  MMMM  OO      OO  RR      RR  YY      YY  AA      AA  LL          CC
MM  MM  MM  EE          MM  MM  MM  OO      OO  RR      RR    YY  YY    AA      AA  LL          CC
MM  MM  MM  EE          MM  MM  MM  OO      OO  RR      RR    YY  YY    AA      AA  LL          CC
MM      MM  EEEEEEEE    MM      MM  OO      OO  RRRRRRRR        YY      AA      AA  LL          CC
MM      MM  EEEEEEEE    MM      MM  OO      OO  RRRRRRRR        YY      AA      AA  LL          CC
MM      MM  EE          MM      MM  OO      OO  RR  RR          YY      AAAAAAAAAA  LL          CC
MM      MM  EE          MM      MM  OO      OO  RR  RR          YY      AAAAAAAAAA  LL          CC
MM      MM  EE          MM      MM  OO      OO  RR    RR        YY      AA      AA  LL          CC          ....
MM      MM  EE          MM      MM  OO      OO  RR    RR        YY      AA      AA  LL          CC          ....
MM      MM  EEEEEEEEEE  MM      MM    OOOOOO    RR      RR      YY      AA      AA  LLLLLLLLLL    CCCCCCCC  ....
MM      MM  EEEEEEEEEE  MM      MM    OOOOOO    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            .TITLE  MEMORYALC - DYNAMIC MEMORY ALLOCATION
0000            .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ; D. N. CUTLER 3-AUG-76
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-031 WMC0011         Wayne Cardoza           27-Aug-1984
0000    ;               Return correct status after pool expansion fails.
0000    ;
0000    ;       V03-030 WMC0010         Wayne Cardoza           23-Aug-1984
0000    ;               Output message on pool expansion failure.
0000    ;
0000    ;       V03-029 DAS0002         David Solomon           30-Apr-1984
0000    ;               Fix broken branch destination.
0000    ;
0000    ;       V03-028 DAS0001         David Solomon           30-Apr-1984
0000    ;               Fix broken word reference to IOC$GL_xxx.
0000    ;
0000    ;       V03-027 CWH3027         CW Hobbs                29-Apr-1984
0000    ;               Fix broken branch.
0000    ;
0000    ;       V03-026 EMD0072         Ellen M. Dusseault      6-Apr-1984
0000    ;               Fix broken branches.
0000    ;
0000    ;       V03-025 KPL0003         Peter Lieberwirth       31-Mar-1984
0000    ;               Fix register useage bug in EXE$ALOPHYCNTG.  Fix some
0000    ;               broken branches.
0000    ;
0000    ;       V03-024 SRB0118         Steve Beckhart          26-Mar-1984
0000    ;               Fixed broken branches.
0000    ;
0000    ;       V03-024 KDM0095         Kathleen D. Morse       23-Mar-1984
```

```
0000  ;                 Maximize size of slave CEB with regular CEB in EXE$ALLOCCEB.
0000  ;
0000  ;         V03-023 WMC0009             Wayne Cardoza               08-Mar-1984
0000  ;                 Add entry point for non-paged variable length pool.
0000  ;
0000  ;         V03-022 WMC0008             Wayne Cardoza               25-Feb-1984
0000  ;                 Optimize EXE$ALLOCBUF
0000  ;
0000  ;         V03-021 KPL0002             Peter Lieberwirth           22-Feb-1984
0000  ;                 Fix bugs in V03-019, specifically: move routine to
0000  ;                 non-paged PSECT, correct pointers that are zero-based.
0000  ;
0000  ;         V03-020 LY00B3              Larry Yetto                 10-FEB-1984 10:10
0000  ;                 Fix truncation errors
0000  ;
0000  ;         V03-019 KPL0001             Peter Lieberwirth           5-Feb-1984
0000  ;                 Add EXE$ALOPHYCNTG routine that allocates physically
0000  ;                 contiguous memory and maps it in system virtual address
0000  ;                 space.
0000  ;
0000  ;         V03-018 LJK0257             Lawrence J. Kenah           6-Jan-1984
0000  ;                 Add EXE$ALOPAGWAIT routine that parallels a similar for
0000  ;                 nonpaged pool. Remove EXE$ALLOCPQB entry point.
0000  ;
0000  ;         V03-017 DWT0157             David W. Thiel              29-Dec-1983
0000  ;                 Fix broken branches.
0000  ;
0000  ;         V03-016 TMK0001             Todd M. Katz                13-Nov-1983
0000  ;                 Fix a broken branch.
0000  ;
0000  ;         V03-015 CWH3015             CW Hobbs                    29-Oct-1983
0000  ;                 Fix broken branch.
0000  ;
0000  ;         V03-014 DWT0123             David W. Thiel              23-Aug-1983
0000  ;                 Improve performance of main path through EXE$ALLOCATE by
0000  ;                 deleting the calculation of the largest free block.
0000  ;                 Improve performance of EXE$DEANONPGDSIZ by avoiding
0000  ;                 call to SCH$RAVAIL on returning a block to a lookaside
0000  ;                 list when the lookaside list was not previously empty.
0000  ;                 Add EXE$DEAPGDSIZ entry point to deallocate an
0000  ;                 arbirtrary quantity of paged dynamic memory.
0000  ;
0000  ;         V03-013 SRB0098             Steve Beckhardt             19-Jul-1983
0000  ;                 Fixed bug in extending pool.  Changed EXE$ALONONPAGED
0000  ;                 to return SS$_INSFMEM on error.
0000  ;
0000  ;         V03-012 SRB0076             Steve Beckhardt             8-Apr-1983
0000  ;                 Modified EXE$ALONONPAGED to preserve R3.  Added entry point
0000  ;                 EXE$ALONPAGWAITS which is similar to EXE$ALONPAGWAIT but
0000  ;                 may be called at IPL$_SYNCH.
0000  ;
0000  ;         V03-011 ROW0178             Ralph O. Weber              5-APR-1983
0000  ;                 Enhance the "cannot expand pool right now because IPL is
0000  ;                 wrong" test in EXE$EXTENDPOOL to allow pool expansion when the
0000  ;                 current IPL is at or below IPL$_SYNCH or the processor is not
0000  ;                 running on the interrupt stack.
0000  ;
```

```
0000        ;           V03-010 RSH0006           R. Scott Hanna            28-Feb-1983
0000        ;                   Temporary disable of size and address granularity check
0000        ;                   in EXE$DEAP1. This check should be restored when all users
0000        ;                   of the P1 allocation region call EXE$ALOP1PROC or
0000        ;                   EXE$ALPO1IMAG to allocate space.
0000        ;
0000        ;           V03-009 JWH0183           Jeffrey W. Horn           10-Feb-1983
0000        ;                   Modify P1 routines to round requests to proper allocation
0000        ;                   boundries.
0000        ;
0000        ;           V03-008 STJ3052           Steven Jeffreys           20-Jan-1983
0000        ;                   Changed W^ to L^ to fix link truncation error.
0000        ;
0000        ;           V03-007 WMC0007           Wayne Cardoza             10-Jan-1983
0000        ;                   Fix MOVZBL of PCB length (grew past byte length).
0000        ;
0000        ;           V03-006 JWH0142           Jeffrey W. Horn           20-Nov-1982
0000        ;                   Optimize routines added in JWH0119.  Also allow these
0000        ;                   routines to be called from elevated IPL.
0000        ;
0000        ;           V03-005 JWH0119           Jeffrey W. Horn            3-Nov-1982
0000        ;                   Add  EXE$ALOP1IMAG, EXE$ALOP1PROC, and EXE$DEAP1
0000        ;                   routines.
0000        ;
0000        ;           V03-004 CWH0001           CW Hobbs                  19-Aug-1982
0000        ;                   Add call to $PFNDEF macro
0000        ;
0000        ;           V03-003 WMC0001           Wayne Cardoza             31-Jul-1982
0000        ;                   Fill in PFN data base when expanding non-paged pool.
0000        ;
0000        ;           V03-002 SRB0052           Steve Beckhardt            2-Jun-1982
0000        ;                   Modified EXE$DEALLOCATE to detect overlapping deallocates
0000        ;                   and bugcheck if this occurs.
0000        ;
0000        ; DYNAMIC MEMORY ALLOCATION
0000        ;
0000        ; MACRO LIBRARY CALLS
0000        ;
0000
0000                $CEBDEF                         ;DEFINE COMMON EVENT BLOCKS
0000                $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
0000                $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
0000                $IRPDEF                         ;DEFINE IRP OFFSETS
0000                $IMPDEF                         ;DEFINE RMS IMPURE AREA SYMBOLS
0000                $JIBDEF                         ;DEFINE JIB OFFSETS
0000                $PCBDEF                         ;DEFINE PCB OFFSETS
0000                $PFNDEF                         ;DEFINE PFN CONSTANTS & OFFSETS
0000                $PHDDEF                         ;DEFINE PHD OFFSETS
0000                $PQBDEF                         ;DEFINE PQB OFFSETS
0000                $PRDEF                          ;DEFINE PROCESSOR REGISTERS
0000                $PSLDEF                         ;DEFINE PSL FIELDS
0000                $PTEDEF                         ;DEFINE PTE FIELDS
0000                $RSNDEF                         ;DEFINE RESOURCE WAIT NUMBERS
0000                $SHBDEF                         ;DEFINE SHARED MEM CONTROL BLOCK
0000                $SHDDEF                         ;DEFINE SHARED MEM DATAPAGE
0000                $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
0000                $TQEDEF                         ;DEFINE TQE OFFSETS
```

```
                  0000          $TTYDEF                         ;TWP
                  0000          $VADEF                          ;DEFINE VIRTUAL ADDRESS FIELDS
                  0000
                  0000  ;
                  0000  ; LOCAL SYMBOLS
                  0000  ;
                  0000  ; ALLOCATION GRANULARITY MASK
                  0000  ;
                  0000
0000000F          0000  MASK=^XF                                ;16 BYTE ALLOCATION GRANULARITY
0000000F          0000  EXE$C_ALCGRNMSK==MASK                   ; ALLOCATION GRANULARITY MASK
                  0000
         00000000               .PSECT  AEXENONPAGED            ; NONPAGED EXEC
```

```
0000         .SBTTL  ALLOCATE MEMORY AND CONDITIONALLY WAIT
0000   ;+
0000   ; EXE$ALLOCBUF - ALLOCATE BUFFERED I/O BUFFER AND CONDITIONALLY WAIT
0000   ;
0000   ; THIS ROUTINE IS CALLED TO ALLOCATE A BUFFERED I/O BUFFER. IF SUFFICIENT
0000   ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
0000   ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000   ;
0000   ; EXE$ALLOCCEB - ALLOCATE COMMON EVENT BLOCK AND CONDITIONALLY WAIT
0000   ;
0000   ; THIS ROUTINE IS CALLED TO ALLOCATE A COMMON EVENT BLOCK. IF SUFFICIENT
0000   ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
0000   ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000   ;
0000   ; EXE$ALLOCJIB - ALLOCATE JOB INFORMATION BLOCK AND CONDITIONALLY WAIT
0000   ;
0000   ; THIS ROUTINE IS CALLED TO ALLOCATE A JOB INFORMATION BLOCK. IF SUFFICIENT
0000   ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED
0000   ; DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000   ;
0000   ; EXE$ALLOCIRP - ALLOCATE I/O REQUEST PACKET AND CONDITIONALLY WAIT
0000   ;
0000   ; THIS ROUTINE IS CALLED TO ALLOCATE AN I/O PACKET. IF SUFFICIENT MEMORY
0000   ; IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED
0000   ; DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000   ;
0000   ; EXE$ALLOCPCB - ALLOCATE PROCESS CONTROL BLOCK AND CONDITIONALLY WAIT
0000   ;
0000   ; THIS ROUTINE IS CALLED TO ALLOCATE A PROCESS CONTROL BLOCK WHEN
0000   ; CREATING A NEW PROCESS.  IF SUFFICIENT MEMORY IS NOT AVAILABLE, THEN
0000   ; A RESOURCE WAIT STATE IS CONDITIONALLY ENTERED DEPENDING ON THE CURRENT
0000   ; PROCESS' RESOURCE WAIT MODE.
0000   ;
0000   ; EXE$ALLOCPQB - ALLOCATE PROCESS QUOTA BLOCK AND CONDITIONALLY WAIT
0000   ;
0000   ; THIS ROUTINE IS CALLED TO ALLOCATE A PROCESS QUOTA BLOCK WHEN CREATING
0000   ; A NEW PROCESS.  IF SUFFICIENT MEMORY IS NOT AVAILABLE, THEN A RESOURCE
0000   ; WAIT STATE IS ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT
0000   ; MODE.
0000   ;
0000   ; EXE$ALLOCTQE - ALLOCATE TIME QUEUE ENTRY AND CONDITIONALLY WAIT
0000   ;
0000   ; THIS ROUTINE IS CALLED TO ALLOCATE A TIME QUEUE ENTRY. IF SUFFICIENT
0000   ; MEMORY IS NOT AVAILABLE, THEN A RESOURCE WAIT STATE IS CONDITIONALLY
0000   ; ENTERED DEPENDING ON THE CURRENT PROCESS' RESOURCE WAIT MODE.
0000   ;
0000   ; INPUTS:
0000   ;
0000   ;       R4 = NORMALLY CURRENT PROCESS PCB ADDRESS, BUT NOT REQUIRED.
0000   ;
0000   ; IF ENTRY AT EXE$ALLOCBUF, THEN
0000   ;
0000   ;       R1 = SIZE OF REQUESTED BUFFER IN BYTES.
0000   ;
0000   ; OUTPUTS:
0000   ;
0000   ;       R0 = LOW BIT CLEAR IF ALLOCATION FAILURE WITH CALLING IPL PRESERVED.
```

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

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

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

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

```
                             00EA         .SBTTL  ALLOCATE NONPAGED DYNAMIC MEMORY
                             00EA  ;+
                             00EA  ; EXE$ALONONPAGED - ALLOCATE NONPAGED DYNAMIC MEMORY
                             00EA  ;
                             00EA  ; EXE$ALONPAGVAR  - ALLOCATE NONPAGED DYNAMIC MEMORY FROM VARIABLE AREA ONLY
                             00EA  ;
                             00EA  ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE NONPAGED POOL.
                             00EA  ; IF THE BLOCK IS THE SAME SIZE AS AN I/O PACKET, AN ATTEMPT IS MADE TO ALLO-
                             00EA  ; CATE IT FROM THE LOOKASIDE LIST.
                             00EA  ;
                             00EA  ; INPUTS:
                             00EA  ;
                             00EA  ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                             00EA  ;
                             00EA  ; OUTPUTS:
                             00EA  ;
                             00EA  ;       R0 = SS$_INSFMEM IF MEMORY IS NOT AVAILABLE.
                             00EA  ;
                             00EA  ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                             00EA  ;
                             00EA  ;               R1 = SIZE OF ALLOCATED BLOCK.
                             00EA  ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                             00EA  ;
                             00EA  ;       OTHER REGISTERS PRESERVED
                             00EA  ;-
                             00EA
                             00EA         .ENABL  LSB
              00C6     31    00EA  200$:  BRW     20$                           ;BAD ALLOCATION REQUEST
                             00ED  EXE$ALONONPAGED::                            ;ALLOCATE NONPAGED MEMORY
 51     000000D0 8F    D1    00ED         CMPL    #<IRP$C_LENGTH+MASK>&<^C<MASK>>,R1  ;SIZE GREATER THAN IRP ?
                16     1F    00F4         BLSSU   LRP                           ;IF LSSU, YES
00000000'EF     51     D1    00F6         CMPL    R1, L^IOC$GL_IRPMIN           ;IS THE BLOCK TOO SMALL?
                64     1F    00FD         BLSSU   SRP                           ;YES, TRY SMALL PACKETS
 52     00000000'FF    0F    00FF         REMQUE  @L^IOC$GL_IRPFL,R2            ;REMOVE FIRST PACKET FROM LOOK ASIDE LIST
                23     1D    0106         BVS     LISTCHK                       ;IF VS EMPTY LIST
           50   01     D0    0108         MOVL    #SS$_NORMAL,R0                ;SET SUCCESSFUL COMPLETION
                05           010B         RSB                                   ;
                             010C
 51     00000000'EF    D1    010C  LRP:   CMPL    L^IOC$GL_LRPMIN,R1            ;SIZE LESS THAN LRP MINIMUM ?
                1C     1A    0113         BGTRU   EXE$ALONPAGVAR                ;IF GTRU, YES
 51     00000000'EF    D1    0115         CMPL    L^IOC$GL_LRPSIZE,R1           ;SIZE GREATER THAN LRP ?
                13     1F    011C         BLSSU   EXE$ALONPAGVAR                ;IF LSSU, YES
 52     00000000'FF    0F    011E         REMQUE  @L^IOC$GL_LRPFL,R2            ;REMOVE FIRST PACEKT FROM LRP LIST
                04     1D    0125         BVS     LISTCHK                       ;IF VS, EMPTY LIS
           50   01     D0    0127         MOVL    #SS$_NORMAL,R0
                05           012A         RSB
                             012B  LISTCHK:
              021A     30    012B         BSBW    EXE$EXTENDPOOL                ;ATTEMPT TO EXTEND POOL
             BC 50     E8    012E         BLBS    R0,EXE$ALONONPAGED            ;RETRY LISTS IF SOMETHING EXTENDED
                             0131  EXE$ALONPAGVAR::
           51   0F     C0    0131         ADDL    #MASK,R1                      ;ROUND SIZE UP TO NEXT BOUNDRY
           51   0F     CA    0134         BICL    #MASK,R1                      ;TRUNCATE SIZE BACK TO MULTIPLE
                B1     13    0137         BEQL    200$                          ;IF EQL BAD ALLOCATION REQUEST
                53     DD    0139         PUSHL   R3                            ;SAVE R3
     53     0000'CF    9E    013B         MOVAB   W^EXE$GL_NONPAGED,R3          ;GET ADDRESS OF NONPAGED MEMORY LISTHEAD
                             0140         DSBINT  (R3)+                         ;DISABLE INTERRUPTS
                72     10    0146         BSBB    EXE$ALLOCATE                  ;ALLOCATE BLOCK
```

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

```
                      0180            .SBTTL  ALLOCATE PAGED DYNAMIC MEMORY
                      0180    ;+
                      0180    ; EXE$ALOPAGED - ALLOCATE PAGED DYNAMIC MEMORY
                      0180    ;
                      0180    ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE PAGED POOL.
                      0180    ;
                      0180    ; INPUTS:
                      0180    ;
                      0180    ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                      0180    ;
                      0180    ; OUTPUTS:
                      0180    ;
                      0180    ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                      0180    ;
                      0180    ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                      0180    ;
                      0180    ;               R1 = SIZE OF ALLOCATED BLOCK.
                      0180    ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                      0180    ;-
                      0180
                      0180    EXE$ALOPAGED::                                  ;ALLOCATE PAGED DYNAMIC MEMORY
        51    0F  C0  0180            ADDL    #MASK,R1                        ;ROUND SIZE UP TO NEXT BOUNDRY
        51    0F  CA  0183            BICL    #MASK,R1                        ;TRUNCATE SIZE BACK TO MULTIPLE
              2B  13  0186            BEQL    20$                             ;IF EQL BAD ALLOCATION REQUEST
                      0188            SAVIPL                                  ;SAVE CURRENT IPL
              54  DD  018B            PUSHL   R4                              ;SAVE REGISTER
   50     0000'CF 9E  018D            MOVAB   W^EXE$GL_PGDYNMTX,R0            ;GET ADDRESS OF PAGED MEMORY MUTEX
54    00000000'EF D0  0192            MOVL    L^SCH$GL_CURPCB,R4              ;GET CURRENT PROCESS PCB ADDRESS
           FE64'  30  0199            BSBW    SCH$LOCKW                       ;LOCK PAGED MEMORY DATA BASE FOR WRITE
   53     0000'CF 9E  019C            MOVAB   W^EXE$GL_PAGED,R3               ;GET ADDRESS OF PAGED MEMORY LISTHEAD
              17  10  01A1            BSBB    EXE$ALLOCATE                    ;ALLOCATE BLOCK
              07  BB  01A3            PUSHR   #^M<R0,R1,R2>                   ;SAVE REGISTERS
   50     0000'CF 9E  01A5            MOVAB   W^EXE$GL_PGDYNMTX,R0            ;GET ADDRESS OF PAGED MEMORY MUTEX
           FE53'  30  01AA            BSBW    SCH$UNLOCK                      ;UNLOCK PAGED MEMORY DATA BASE
              17  BA  01AD            POPR    #^M<R0,R1,R2,R4>                ;RESTORE REGISTERS
                      01AF            ENBINT                                  ;ENABLE INTERRUPTS
                  05  01B2            RSB                                     ;
                      01B3    20$:    BUG_CHECK BADALORQSZ                    ;BAD ALLOCATION REQUEST SIZE
              50  D4  01B7            CLRL    R0                              ;INDICATE NO BLOCK ALLOCATED
                  05  01B9            RSB                                     ;
                      01BA            .DSABL  LSB
```

```
                        01BA         .SBTTL  GENERAL ALLOCATE MEMORY SUBROUTINE
                        01BA  ;+
                        01BA  ; EXE$ALLOCATE - ALLOCATE MEMORY SUBROUTINE
                        01BA  ;
                        01BA  ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM A POOL WHOSE ENTRIES
                        01BA  ; ARE MAINTAINED IN A MEMORY ORDER SORTED LIST.
                        01BA  ;
                        01BA  ; INPUTS:
                        01BA  ;
                        01BA  ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                        01BA  ;       R3 = ADDRESS OF ALLOCATION REGION LISTHEAD.
                        01BA  ;
                        01BA  ; OUTPUTS:
                        01BA  ;
                        01BA  ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                        01BA  ;
                        01BA  ;               R2 = 0 (USED TO BE THE SIZE OF LARGEST BLOCK FOUND)
                        01BA  ;
                        01BA  ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                        01BA  ;
                        01BA  ;               R1 = SIZE OF ALLOCATED BLOCK.
                        01BA  ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                        01BA  ;-
                        01BA
                        01BA  EXE$ALLOCATE::                                  ;ALLOCATE MEMORY
         50    53   D0  01BA         MOVL    R3,R0                            ;COPY ADDRESS OF FIRST FREE BLOCK ADDRESS
         52    50   D0  01BD  10$:   MOVL    R0,R2                            ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
         50    62   D0  01C0         MOVL    (R2),R0                          ;GET ADDRESS OF NEXT FREE BLOCK
               1D   13  01C3         BEQL    30$                              ;IF EQL NO MEMORY AVAILABLE
      04 A0    51   D1  01C5         CMPL    R1,4(R0)                         ;FREE BLOCK BIG ENOUGH?
               F2   1A  01C9         BGTRU   10$                              ;IF GTRU NO
               0E   13  01CB         BEQL    20$                              ;IF EQL FREE BLOCK IS EXACT SIZE
   53    51    50   C1  01CD         ADDL3   R0,R1,R3                         ;CALCULATE ADDRESS OF NEW FREE BLOCK
         83    80   D0  01D1         MOVL    (R0)+,(R3)+                      ;COPY LINK TO NEXT FREE BLOCK
   63    60    51   C3  01D4         SUBL3   R1,(R0),(R3)                     ;CALCULATE SIZE OF NEW FREE BLOCK
         70    73   DE  01D8         MOVAL   -(R3),-(R0)                      ;SET LINK TO NEW FREE BLOCK
         62    60   D0  01DB  20$:   MOVL    (R0),(R2)                        ;COPY LINK TO NEW FREE BLOCK
         52    80   9E  01DE         MOVAB   (R0)+,R2                         ;SET ADR OF ALLOCATED BLOCK, INDICATE SUCCES
                    05  01E1         RSB                                      ;RETURN
                        01E2  ;
                        01E2  ; NO BLOCK OF THE REQUIRED SIZE COULD BE FOUND.  RETURN 0 IN R2 WHERE THE LENGTH OF
                        01E2  ; THE LARGEST FREE BLOCK USED TO GO.
                        01E2  ;
               52   D4  01E2  30$:   CLRL    R2                               ;INITIAL VALUE OF LARGEST FREE BLOCK SEEN
                    05  01E4         RSB                                      ;RETURN FAILURE
```

```
                              01E5         .SBTTL  DEALLOCATE NONPAGED DYNAMIC MEMORY
                              01E5    ;+
                              01E5    ; EXE$DEANONPAGED - DEALLOCATE NONPAGED DYNAMIC MEMORY
                              01E5    ;
                              01E5    ; THIS ROUTINE IS CALLED TO DEALLOCATE A BLOCK OF MEMORY TO A NONPAGED POOL.
                              01E5    ; IF THE BLOCK IS A SHARED MEMORY BLOCK TYPE, THE BLOCK IS DEALLOCATED TO
                              01E5    ; THE SHARED MEMORY POOL.  OTHERWISE, THE BLOCK'S ADDRESS IS CHECKED TO SEE
                              01E5    ; IF IT WAS ALLOCATED FROM THE I/O PACKET LOOKASIDE LIST AND IF SO,
                              01E5    ; IT IS RETURNED TO THAT LIST.  OTHERWISE IT IS MERGED INTO THE NORMAL
                              01E5    ; NONPAGED POOL.
                              01E5    ;
                              01E5    ; INPUT:
                              01E5    ;
                              01E5    ;        IF ENTRY IS AT EXE$DEANONPAGED
                              01E5    ;
                              01E5    ;                RO = ADDRESS OF BLOCK TO BE DEALLOCATED.
                              01E5    ;                IRP$W_SIZE(RO) = SIZE OF BLOCK TO BE DEALLOCATED.
                              01E5    ;                IRP$B_TYPE(RO) = TYPE OF BLOCK TO BE DEALLOCATED.
                              01E5    ;
                              01E5    ;        IF ENTRY IS AT EXE$DEANONPAGED_SIZE
                              01E5    ;
                              01E5    ;                RO = ADDRESS OF BLOCK TO BE DEALLOCATED.
                              01E5    ;                R1 = SIZE OF BLOCK TO BE DEALLOCATED (ASSUMED NONZERO)
                              01E5    ;
                              01E5    ; OUTPUTS:
                              01E5    ;
                              01E5    ;        THE SPECIFIED BLOCK IS RETURNED TO THE APPROPRIATE POOL.
                              01E5    ;-
                              01E5
                              01E5         .ENABL  LSB
                              01E5    EXE$DEANONPAGED::                          ;DEALLOCATE NONPAGED DYNAMIC MEMORY
              0A A0     95    01E5         TSTB    IRP$B_TYPE(RO)              ;IS BLOCK A SHARED MEMORY BLOCK?
                 03     18    01E8         BGEQ    5$                          ;IF GEQ NO
               011C     31    01EA         BRW     EXE$DEASHARED               ;ELSE, DEALLOCATE IT
                              01ED
         51   08 A0     3C    01ED    5$:  MOVZWL  IRP$W_SIZE(RO),R1           ;GET SIZE OF BLOCK IN BYTES
                              01F1
                              01F1    EXE$DEANONPGDSIZ::                         ;DEALLOCATE NONPAGED DYNAMIC MEMORY
            50   0F     D3    01F1         BITL    #MASK,RO                    ;CHECK PACKET ALIGNMENT
                 37     12    01F4         BNEQ    15$                         ;BRANCH ON ERROR -- LET STD. RTN HANDLE IT
     00000000'EF 50     D1    01F6         CMPL    RO,L^IOC$GL_SRPSPLIT        ;SMALL REQUEST PACKET?
                 0A     1F    01FD         BLSSU   8$                          ;BR IF NOT
     00000000'FF 60     0E    01FF         INSQUE  (RO),@L^IOC$GL_SRPBL        ;INSERT NEW PACKET AT END OF LIST
                 36     13    0206         BEQL    20$                         ;BRANCH IF LIST WAS EMPTY, DECLARE RES. AVL.
                        05    0208         RSB                                 ;RSB
                              0209
        0000'CF  50     D1    0209    8$:  CMPL    RO,W^EXE$GL_SPLITADR        ;I/O REQUEST PACKET
                 0A     1F    020E         BLSSU   10$                         ;IF LSSU NO
     00000000'FF 60     0E    0210         INSQUE  (RO),@L^IOC$GL_IRPBL        ;INSERT NEW PACKET AT END OF LIST
                 25     13    0217         BEQL    20$                         ;BRANCH IF LIST WAS EMPTY, DECLARE RES. AVL.
                        05    0219         RSB                                 ;RETURN
                              021A
     00000000'EF 50     D1    021A    10$: CMPL    RO,L^IOC$GL_LRPSPLIT        ;LARGE REQUEST PACKET
                 0A     1F    0221         BLSSU   15$                         ;IF LSSU, NO
     00000000'FF 60     0E    0223         INSQUE  (RO),@L^IOC$GL_LRPBL        ;INSERT NEW PACKET AT END OF LIST
                 12     13    022A         BEQL    20$                         ;BRANCH IF LIST WAS EMPTY, DECLARE RES. AVL.
                        05    022C         RSB                                 ;RETURN
```

```
                    022D   651
          53   10   022D   652 15$:     BSBB    CHECKBLOCK              ;CHECK DEALLOCATION PARAMETERS
  53  0000'CF  9E   022F   653          MOVAB   W^EXE$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 DECLARE 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          .SBTTL  DEALLOCATE PAGED DYNAMIC MEMORY
                              0249  ;+
                              0249  ; EXE$DEAPAGED - DEALLOCATE PAGED DYNAMIC MEMORY
                              0249  ;
                              0249  ; THIS ROUTINE IS CALLED TO DEALLOCATE A BLOCK OF MEMORY TO THE PAGED POOL.
                              0249  ;
                              0249  ; INPUTS:
                              0249  ;
                              0249  ;       IF ENTRY IS AT EXE$DEAPAGED
                              0249  ;
                              0249  ;               R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                              0249  ;               IRP$W_SIZE(R0) = SIZE OF BLOCK TO BE DEALLOCATED.
                              0249  ;
                              0249  ;       IF ENTRY IS AT EXE$DEAPGDSIZ
                              0249  ;
                              0249  ;               R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                              0249  ;               R1 = SIZE OF BLOCK TO BE DEALLOCATED (ASSUMED NONZERO).
                              0249  ;
                              0249  ; OUTPUTS:
                              0249  ;
                              0249  ;       THE SPECIFIED BLOCK OF MEMORY IS RETURNED TO THE PAGED POOL.
                              0249  ;-
                              0249
                              0249  EXE$DEAPAGED::                                  ;DEALLOCATE PAGED DYNAMIC MEMORY
         51    08 A0      3C  0249          MOVZWL  IRP$W_SIZE(R0),R1               ;GET SIZE OF BLOCK IN BYTES
                              024D
                              024D  EXE$DEAPGDSIZ::
               33         10  024D          BSBB    CHECKBLOCK                      ;CHECK DEALLOCATION PARAMETERS
                              024F          SAVIPL                                  ;SAVE CURRENT IPL
               11         BB  0252          PUSHR   #^M<R0,R4>                      ;SAVE REGISTERS
         50    0000'CF    9E  0254          MOVAB   W^EXE$GL_PGDYNMTX,R0            ;GET ADDRESS OF PAGED MEMORY MUTEX
   54    00000000'EF      D0  0259          MOVL    L^SCH$GL_CURPCB,R4              ;GET CURRENT PROCESS PCB ADDRESS
             FD9D'        30  0260          BSBW    SCH$LOCKW                       ;LOCK PAGED MEMORY DATA BASE FOR WRITE
               50       8ED0  0263          POPL    R0                              ;RESTORE REGISTER
         53    0000'CF    9E  0266          MOVAB   W^EXE$GL_PAGED,R3               ;GET ADDRESS OF PAGED MEMORY LISTHEAD
               29         10  026B          BSBB    EXE$DEALLOCATE                  ;DEALLOCATE BLOCK
         50    0000'CF    9E  026D          MOVAB   W^EXE$GL_PGDYNMTX,R0            ;GET ADDRESS OF PAGED MEMORY MUTEX
             FD8B'        30  0272          BSBW    SCH$UNLOCK                      ;UNLOCK PAGED MEMORY DATA BASE
               54       8ED0  0275          POPL    R4                              ;RESTORE REGISTER
         50    05         3C  0278          MOVZWL  #RSN$_PGDYNMEM,R0               ;SET PAGED DYNAMIC MEMORY RESOURCE NUMBER
             FD82'        30  027B  30$:    BSBW    SCH$RAVAIL                      ;MARK RESOURCE AVAILABLE
                              027E          ENBINT                                  ;ENABLE INTERRUPTS
                          05  0281          RSB                                     ;
                              0282          .DSABL  LSB
```

```
                0282        .SBTTL  CHECK BLOCK PARAMETERS SUBROUTINE
                0282   ;
                0282   ; CHECKBLOCK - CHECK BLOCK PARAMETERS SUBROUTINE
                0282   ;
                0282   ; INPUT PARAMETERS:
                0282   ;
                0282   ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED
                0282   ;       R1 = SIZE OF BLOCK TO BE DEALLOCATED
                0282   ;
                0282   ; OUTPUT PARAMETER:
                0282   ;
                0282   ;       R1 = MODIFIED SIZE OF BLOCK TO BE DEALLOCATED
                0282   ;            (SIZE IS ROUNDED UP TO NEXT MULTIPLE OF QUANTUM OF ALLOCATION)
                0282   ;
                0282
                0282   CHECKBLOCK:                             ;CHECK BLOCK PARAMETERS
   50   0F   D3 0282           BITL    #MASK,R0                ;BLOCK ALIGNED ON BOUNDRY?
        08   12 0285           BNEQ    10$                     ;IF NEQ NO - BAD DEALLOCATION
   51   0F   C0 0287           ADDL    #MASK,R1                ;ROUND SIZE UP TO NEXT BOUNDRY
   51   0F   CA 028A           BICL    #MASK,R1                ;TRUNCATE SIZE BACK TO MULTIPLE
        06   12 028D           BNEQ    20$                     ;IF NEQ OKAY
                028F   10$:    BUG_CHECK BADDALRQSZ            ;BAD DEALLOCATION REQUEST SIZE OR ADDRESS
        8E   D5 0293           TSTL    (SP)+                   ;REMOVE RETURN FROM STACK
             05 0295   20$:    RSB                             ;
```

```
                         0296        .SBTTL  GENERAL DEALLOCATION SUBROUTINE
                         0296   ;+
                         0296   ; EXE$DEALLOCATE - DEALLOCATION SUBROUTINE
                         0296   ;
                         0296   ; INPUTS:
                         0296   ;
                         0296   ;        R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                         0296   ;        R1 = SIZE OF BLOCK IN BYTES
                         0296   ;        R3 = ADDRESS OF ALLOCATION REGION LISTHEAD.
                         0296   ;
                         0296   ; OUTPUTS:
                         0296   ;
                         0296   ;        NONE
                         0296   ;-
                         0296
                         0296   EXE$DEALLOCATE::                                ;DEALLOCATE BLOCK
               54   DD   0296            PUSHL   R4                    ;SAVE REGISTERS
               53   DD   0298            PUSHL   R3
          52   53   D0   029A   10$:     MOVL    R3,R2                 ;SAVE ADDRESS OF PREVIOUS FREE BLOCK
          53   62   D0   029D            MOVL    (R2),R3               ;GET ADDRESS OF NEXT FREE BLOCK
               07   13   02A0            BEQL    20$                   ;IF EQL END OF LIST
          53   50   D1   02A2            CMPL    R0,R3                 ;BLOCK LOGICALLY GO HERE?
               F3   1A   02A5            BGTRU   10$                   ;IF GTRU NO
               3F   13   02A7            BEQLU   50$                   ;IF EQLU DOUBLE DEALLOCATION
          60   53   D0   02A9   20$:     MOVL    R3,(R0)               ;ASSUME NO AGGLOMERATION
               11   13   02AC            BEQL    30$                   ;END OF LIST - NO AGGLOMERATION
     54   51   50   C1   02AE            ADDL3   R0,R1,R4              ;CALCULATE ADDRESS OF END OF BLOCK
          54   53   D1   02B2            CMPL    R3,R4                 ;END OF BLOCK EQUAL TO NEXT IN LIST?
               08   1A   02B5            BGTRU   30$                   ;IF GTR DO NOT AGGLOMERATE
               2F   1F   02B7            BLSSU   50$                   ;IF LSS OVERLAPPING DEALLOCATE
          60   83   D0   02B9            MOVL    (R3)+,(R0)            ;MOVE LINK TO BLOCK BEING RELEASED
          51   63   C0   02BC            ADDL    (R3),R1               ;ACCUMULATE LENGTH OF NEW FREE BLOCK
          54   52   D0   02BF   30$:     MOVL    R2,R4                 ;CALCULATE ENDING ADDRESS OF PREVIOUS BLOCK
          82   50   D0   02C2            MOVL    R0,(R2)+              ;ASSUME NO AGGLOMERATION
          54   62   C0   02C5            ADDL    (R2),R4               ;ADD LENGTH TO BLOCK BASE ADDRESS
          54   50   D1   02C8            CMPL    R0,R4                 ;END ADDRESS EQUAL TO BLOCK BEING RELEASED?
               0B   1A   02CB            BGTRU   40$                   ;IF GTR DO NOT AGGLOMERATE
               11   1F   02CD            BLSSU   45$                   ;IF LSS MAY BE OVERLAPPING DEALLOCATE
          51   62   C0   02CF            ADDL    (R2),R1               ;ACCUMULATE SIZE OF NEW FREE BLOCK
          72   60   D0   02D2            MOVL    (R0),-(R2)            ;MOVE LINK TO PREVIOUS FREE BLOCK
          50   52   D0   02D5            MOVL    R2,R0                 ;SET ADDRESS OF NEW FREE BLOCK
       04 A0   51   D0   02D8   40$:     MOVL    R1,4(R0)              ;SET SIZE OF FREE BLOCK
          53   8E   7D   02DC            MOVQ    (SP)+,R3              ;RESTORE REGISTERS
                    05   02DF            RSB
                         02E0
                         02E0   45$:     ; IF WE COME HERE IT IS EITHER AN OVERLAPPING DEALLOCATE OR R2
                         02E0            ; (THE PREVIOUS BLOCK POINTER) IS POINTING TO THE LIST HEAD.
                         02E0
          52   04   C2   02E0            SUBL    #4,R2                 ;BACK UP R2
          6E   52   D1   02E3            CMPL    R2,(SP)               ;IS IT POINTING TO THE LIST HEAD?
               F0   13   02E6            BEQL    40$                   ;YES, RESUME IN NORMAL PATH
                         02E8
                         02E8   50$:     BUG_CHECK DOUBLDEALO,FATAL    ;DOUBLE DEALLOCATION OF MEMORY BLOCK
                         02EC
```

```
                         02EC         .SBTTL  ALLOCATE A BLOCK OF SHARED MEMORY POOL
                         02EC     ;+
                         02EC     ; EXE$ALOSHARED - ALLOCATE A BLOCK OF SHARED MEMORY POOL
                         02EC     ;
                         02EC     ; THIS ROUTINE IS CALLED TO ALLOCATE A BLOCK OF MEMORY FROM THE
                         02EC     ; SHARED MEMORY POOL.
                         02EC     ;
                         02EC     ; INPUTS:
                         02EC     ;
                         02EC     ;       R2 = ADDRESS OF SHARED MEMORY CONTROL BLOCK (SHB).
                         02EC     ;
                         02EC     ; OUTPUTS:
                         02EC     ;
                         02EC     ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                         02EC     ;
                         02EC     ;       R0 = LOW BIT SET IF MEMORY ALLOCATED WITH:
                         02EC     ;
                         02EC     ;               R1 = SIZE OF ALLOCATED BLOCK.
                         02EC     ;               R2 = ADDRESS OF ALLOCATED BLOCK.
                         02EC     ;-
                         02EC
                         02EC     EXE$ALOSHARED::                                   ;ALLOCATE SHARED MEMORY POOL
      50   04 A2    D0   02EC             MOVL    SHB$L_DATAPAGE(R2),R0     ;GET ADDRESS OF DATA PAGE
                         02F0                                               ;(PAGE ALIGNED SO LOW BIT CLEAR)
              51    D4   02F0             CLRL    R1                        ;INIT RETRY COUNT
                         02F2
     52    00F8 C0  5E   02F2     10$:    REMQHI  SHD$Q_POOL(R0),R2         ;REMOVE A BLOCK FROM POOL
              09    1F   02F7             BCS     30$                       ;BR IF QUEUE LOCKED - RETRY
              06    1D   02F9             BVS     20$                       ;BR IF NO ENTRY - FAILURE
      51   08 A2    3C   02FB             MOVZWL  IRP$W_SIZE(R2),R1         ;GET SIZE OF BLOCK
              50    D6   02FF             INCL    R0                        ;SET SUCCESSFUL COMPLETION
                    05   0301     20$:    RSB                               ;
                         0302
  EA 51   0000'CF   F3   0302     30$:    AOBLEQ  W^EXE$GL_LOCKRTRY,R1,10$ ;INCREMENT RETRY COUNT AND TRY AGAIN
                    05   0308             RSB                               ;IF RETRIES EXHAUSTED, QUEUE HEADER BAD
```

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

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

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

```
                           045C          .SBTTL  EXTENDLIST - EXTEND SELECTED PACKET LIST
                           045C  ;+
                           045C  ; EXTENDLIST - EXTEND SELECTED PACKET LIST IF NEEDED
                           045C  ;
                           045C  ; THIS ROUTINE EXTENDS THE SELECTED LIST IF NECESSARY BY ADDING ONE
                           045C  ; PAGE AT A TIME
                           045C  ;
                           045C  ; INPUTS:
                           045C  ;
                           045C  ;         R2 = PACKET SIZE
                           045C  ;         R3 = ADDRESS OF MMG$GL_xRPNEXT, NEXT VA TO ALLOCATE
                           045C  ;         R6 = ADDRESS OF IOC$GL_xRPREM, ADDRESS OF PARTIAL PACKET
                           045C  ;         R7 = ADDRESS OF IOC$GL_xRPCNT, COUNT OF ALLOCATED PACKETS
                           045C  ;         R8 = ADDRESS OF IOC$GL_xRPFL, LIST HEADER
                           045C  ;         R9 = ADDRESS OF SGN$GL_xRPCNTV, MAXIMUM COUNT TO ALLOCATE
                           045C  ;
                           045C  ; OUTPUTS:
                           045C  ;
                           045C  ;         R0 = COMPLETION STATUS, LOW BIT CLEAR IF NO EXTENSION OR NOT NEEDED
                           045C  ;
                           045C  ;-
                           045C  EXTENDLIST:                                     ;
          68    58    D1   045C          CMPL    R8,(R8)                         ; IS LIST EMPTY?
                48    12   045F          BNEQ    70$                             ; IF NOT, EXIT
          67    69    D1   0461          CMPL    (R9),(R7)                       ; CHECK FOR POSSIBLE EXTENSION
                43    15   0464          BLEQ    70$                             ; NO, LIST IS AT MAXIMUM SIZE
                44    10   0466          BSBB    EXTENDPAGE                      ; ATTEMPT TO ADD A PAGE
                50    D5   0468          TSTL    R0                              ; CHECK FOR SUCCESS
                3D    19   046A          BLSS    70$                             ; BR IF FAILED TO EXTEND
          50    66    D0   046C          MOVL    (R6),R0                         ; GET ADDRESS OF PARTIAL PACKET
                0A    12   046F          BNEQ    10$                             ; BR IF ONE PRESENT
50    63    00000200 8F    C3   0471          SUBL3   #512,(R3),R0                    ; COMPUTE ADDRESS OF NEW PAGE
                60    D4   0479          CLRL    (R0)                            ; INIT SIZE
          51    60    D0   047B  10$:    MOVL    (R0),R1                         ; GET SIZE OF FRAGMENT
     51   0200    C1    9E   047E          MOVAB   512(R1),R1                      ; AND AUGMENT BY NEW PAGE
          51    52    C2   0483  20$:    SUBL    R2,R1                           ; DIMINISH SIZE BY ONE PACKET
                0E    19   0486          BLSS    30$                             ; BRANCH IF INSUFFICIENT SPACE LEFT
       08 A0    52    B0   0488          MOVW    R2,IRP$W_SIZE(R0)               ; SET SIZE
                67    D6   048C          INCL    (R7)                            ; COUNT ANOTHER PACKET
          68    60    0E   048E          INSQUE  (R0),(R8)                       ; INSERT ON LIST
          50    52    C0   0491          ADDL    R2,R0                           ; ADVANCE ADDRESS
                ED    11   0494          BRB     20$                             ;
          51    52    C0   0496  30$:    ADDL    R2,R1                           ; CORRECT SIZE OF FRAGMENT
                05    13   0499          BEQL    40$                             ; BR IF NO FRAGMENT
          60    51    D0   049B          MOVL    R1,(R0)                         ; SAVE SIZE OF FRAGMENT
                02    11   049E          BRB     50$                             ;
                50    D4   04A0  40$:    CLRL    R0                              ; ZAP FRAGMENT POINTER
          66    50    D0   04A2  50$:    MOVL    R0,(R6)                         ; SAVE NEW FRAGMENT POINTER
          50    01    D0   04A5  60$:    MOVL    #1,R0                           ; SET SUCCESS
                      05   04A8          RSB
                50    D4   04A9  70$:    CLRL    R0                              ; SET FAILURE
                      05   04AB          RSB
                           04AC
```

```
                              04AC        .SBTTL  EXTENDPAGE - EXTEND SPECIFIED AREA BY ONE PAGE
                              04AC  ;+
                              04AC  ; EXTENDPAGE - EXTEND SPECIFIED AREA BY ONE PAGE
                              04AC  ;
                              04AC  ; THIS ROUTINE EXTENDS THE SPECIFIED AREA BY ONE PAGE PROVIDED THAT THE
                              04AC  ; PAGE CAN BE ALLOCATED WITHOUT REDUCING THE FLUID PAGES BELOW THE NUMBER
                              04AC  ; REQUIRED TO ACCOMODATE THE CURRENT MAXIMUM WORKING SET SIZE.  THE EXTENSION
                              04AC  ; MAY ALSO BE PREVENTED IF NO PAGES ARE AVAILABLE OR THE SPECIFIED AREA IS
                              04AC  ; FULL.
                              04AC  ;
                              04AC  ; INPUTS:
                              04AC  ;
                              04AC  ;       R3 = ADDRESS OF CELL CONTAINING VA TO BACK WITH PHYSICAL PAGE
                              04AC  ;
                              04AC  ; OUTPUTS:
                              04AC  ;
                              04AC  ;       R0 = COMPLETION STATUS, NEGATIVE MEANS FAILURE
                              04AC  ;       @R3 = ADVANCED TO POINT TO THE NEXT PAGE IF SUCCESS
                              04AC  ;
                              04AC  ;       R1,R2,R3 PRESERVED
                              04AC  ;-
                              04AC  EXTENDPAGE:
             50     01    CE  04AC          MNEGL   #1,R0                           ; ASSUME FAILURE
                              04AF          DSBINT  L^EXE$GL_NONPAGED               ; SYNCHRONIZE
                    06    BB  04B9          PUSHR   #^M<R1,R2>                      ; SAVE REGISTERS
      52   63       15    09  04BB          EXTZV   #VA$V_VPN,#VA$S_VPN,(R3),R2     ; GET VIRTUAL PAGE NUMBER
         52   0000'DF42    DE  04C0          MOVAL   @W^MMG$GL_SPTBASE[R2],R2        ; COMPUTE SVAPTE
                    62    D5  04C6          TSTL    (R2)                            ; CHECK FOR EMPTY
                    5D    12  04C8          BNEQ    90$                             ; BR IF FILLED
         51   0000'CF     3C  04CA          MOVZWL  W^MPW$GW_LOLIM,R1               ; GET LOW LIMIT FOR MODIFY LIST
         51   0000'CF     C0  04CF          ADDL    W^SGN$GL_FREELIM,R1             ; ADD FREE LIST LIMIT
  51   00000000'EF   51   C3  04D4          SUBL3   R1,L^PFN$GL_PHYPGCNT,R1         ; AND COMPUTE NET FLUID PAGE COUNT
          0000'CF    51   D1  04DC          CMPL    R1,W^SGN$GL_MAXWSCNT            ; COMPARE WITH MAX WORKING SET
                    44    15  04E1          BLEQ    90$                             ; BR IF NO ROOM FOR GROWTH
                    0E    BB  04E3          PUSHR   #^M<R1,R2,R3>                   ; SAVE VOLATILE REGISTERS
                 FB18'    30  04E5          BSBW    MMG$ALLOCPFN                    ; AND ATTEMPT TO ALLOCATE A PFN
                    0E    BA  04E8          POPR    #^M<R1,R2,R3>                   ; RESTORE VOLATILE REGISTERS
                    50    D5  04EA          TSTL    R0                              ; CHECK STATUS
                    39    19  04EC          BLSS    90$                             ; BR IF NO PAGE ALLOCATED
      00000000'EF   01    8A  04EE          BICB    #1,IOC$GL_POOLEXP_STS           ; POOL SUCCESSFULLY EXPANDED
           0000'DF40      B6  04F5          INCW    @W^PFN$AW_REFCNT[R0]            ; REFERENCE COUNT
        0000'DF40   52    D0  04FA          MOVL    R2,@W^PFN$AL_PTE[R0]            ; PTE BACK-POINTER
        0000'DF40   07    90  0500          MOVB    #PFN$C_ACTIVE,@W^PFN$AB_STATE[R0] ; STATE IS ACTIVE
        0000'DF40   01    90  0506          MOVB    #1,@W^PFN$AB_TYPE[R0]           ; TYPE IS SYSTEM PAGE
  62   50   B0000000 8F   C9  050C          BISL3   #<PTE$C_ERKW!PTE$M_VALID>,R0,(R2) ; SET PAGE INTO SPT
            00000000'EF   D7  0514          DECL    L^PFN$GL_PHYPGCNT               ; ONE LESS FLUID PAGE
       63   00000200 8F   C0  051A          ADDL    #512,(R3)                       ; ADVANCE POINTER TO NEXT PAGE
                    06    BA  0521  80$:    POPR    #^M<R1,R2>                      ; RESTORE REGISTERS
                              0523          ENBINT                                  ; RESTORE IPL
                    05        0526          RSB
                              0527  ;
                              0527  ; Pool expansion failure
                              0527  ;
   F2 00000000'EF   00    E2  0527  90$:    BBSS    #0,IOC$GL_POOLEXP_STS,80$       ; MESSAGE ALREADY OUTPUT
   EA 00000000'EF   00    E5  052F          BBCC    #0,IOC$GT_NOPOOL_TWP,80$        ; TWP IN USE
      52   00000030'EF    9E  0537          MOVAB   IOC$GT_NOPOOL_TWP+TTY$K_WB_LENGTH,R2 ; MESSAGE ADDRESS
      51   00000002'EF    3C  053E          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         .SBTTL  ALLOCATE PHYSICALLY-CONTIGUOUS MEMORY
                        0550  ;++
                        0550  ; EXE$ALOPHYCNTG
                        0550  ;
                        0550  ; Allocate and map to system virtual address space N physically-contiguous
                        0550  ; pages.
                        0550  ;
                        0550  ; Inputs:
                        0550  ;         R1 = N = number of physically-contiguous pages
                        0550  ;
                        0550  ; Implicit Inputs:
                        0550  ;         IPL = SYNCH
                        0550  ;
                        0550  ; Outputs:
                        0550  ;         R0 = status: SUCCESS, INSFMEM, INSFSPTS
                        0550  ;         R1 = preserved
                        0550  ;         R2 = system virtual address of N pages of memory if success
                        0550  ;         all other registers preserved
                        0550  ;
                        0550  ; Implicit Outputs:
                        0550  ;         None.
                        0550  ;
                        0550  ; Side Effects:
                        0550  ;         MMG$ALLOCONTIG called
                        0550  ;         IOC$ALLOSPT called - so SPTs are allocated
                        0550  ;
                        0550  ;--
                        0550  EXE$ALOPHYCNTG::
           3A      BB   0550          PUSHR   #^M<R1,R3,R4,R5>         ; save work registers
                        0552                                           ;  r1 = input used as loop counter
                        0552                                           ;  r3 = address of SPT
                        0552                                           ;  r4 = index into PFN database
                        0552                                           ;  r5 = temp storage
  00000000'EF      16   0552          JSB     MMG$ALLOCONTIG           ; find N physically-contiguous pages
                        0558  ;
                        0558  ; MMG$ALLOCONTIG returns:
                        0558  ;         R0 = first PFN in range, or negative if cannot fulfill request
                        0558  ;
       55    50    D0   0558          MOVL    R0,R5                    ; success?
             56    19   055B          BLSS    20$                      ; get out if none found
          FAA0'    30   055D          BSBW    IOC$ALLOSPT              ; allocate N SPTs to map VAs
          57 50    E9   0560          BLBC    R0,30$                   ; if LBC, no system page table slots
                        0563  ;
                        0563  ; IOC$ALLOSPT returns:
                        0563  ;         R1 = preserved, R2 = SVPN (index into SPT), R3 = address of SPT
                        0563  ;
                        0563  ; The main loop indexes backwards through the system page table entries
                        0563  ; and backwards through the PFN database.  It goes backwards so that the
                        0563  ; last system virtual address calculated can be returned to the caller.
                        0563  ;
                        0563                                           ;  r0 = index into SPT
  50    52    51   C1   0563          ADDL3   R1,R2,R0                 ; start at last SPT and go backwards
  54    51    55   C1   0567          ADDL3   R5,R1,R4                 ; start PFNs at end in loop
                        056B  10$:                                     ; set up system page-table entry
             50    D7   056B          DECL    R0                       ; back up SVPN index
             54    D7   056D          DECL    R4                       ; back up PFN index
     6340    54    D0   056F          MOVL    R4,(R3)[R0]              ; fill PFN in SPT
```

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

```
                            00000000          .PSECT  Y$EXEPAGED
                                0000          .SBTTL  *** THE FOLLOWING ROUTINES ARE IN THE PAGED EXEC ***
                                0000          .SBTTL  EXE$ALOP1PROC - ALLOCATE MEMORY FROM PROCESS ALLOCATION REGION
                                0000
                                0000  ;+
                                0000  ; EXE$ALOP1PROC - ALLOCATE MEMORY FROM PROCESS ALLOCATION REGION.
                                0000  ;
                                0000  ; This routine is called to allocate memory from the Process Allocation Region
                                0000  ; for those requests which need memory beyond the life of an image.
                                0000  ;
                                0000  ; INPUTS:
                                0000  ;
                                0000  ;       R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                                0000  ;
                                0000  ; OUTPUTS:
                                0000  ;
                                0000  ;       R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                                0000  ;
                                0000  ;       R0 = LOW BIT SET IF MEMRY ALLOCATED WITH:
                                0000  ;
                                0000  ;               R1 = SIZE OF ALLOCATED BLOCK
                                0000  ;               R2 = ADDRESS OF ALLOCATED BLOCK
                                0000  ;
                                0000  ;       R3 Destroyed.
                                0000  ;
                                0000  ; ENVIRONMENT:
                                0000  ;
                                0000  ;       Called from EXEC or KERNEL modes, this routine will switch
                                0000  ;       into KERNEL mode and disable ASTs to perform the actual
                                0000  ;       allocation.
                                0000  ;
                                0000  ;       IPL =  0 or ASTDEL
                                0000  ;
                                0000  ;--
                                0000
                                0000  EXE$ALOP1PROC::
              51    0F    C0    0000          ADDL    #MASK,R1                        ; round size to next boundry
              51    0F    CA    0003          BICL    #MASK,R1                        ; truncate size back to multiple
                  00E0    30    0006          BSBW    P1SYNCH                         ; get to correct mode, AST level
                                0009
                                0009  ;
                                0009  ; Now in Kernel Mode with ASTs disabled
                                0009  ;
                                0009
  53     00000000'9F      DE    0009          MOVAL   @#CTL$GQ_ALLOCREG,R3            ; get list head address
                 01A7'    30    0010          BSBW    EXE$ALLOCATE                    ; allocate block
                 07 50    E9    0013          BLBC    R0,10$                          ; get out on error
  00000000'9F     51      C2    0016          SUBL    R1,@#CTL$GL_PRCALLCNT           ; keep track of space in use
                    05          001D  10$:    RSB
```

```
                        001E        .SBTTL  EXE$ALOP1IMAG - ALLOCATE MEMORY FROM PROCESS ALLOCATION REGION
                        001E
                        001E  ;+
                        001E  ; EXE$ALOP1IMAG - ALLOCATE MEMORY FROM PROCESS ALLOCATION REGION.
                        001E  ;
                        001E  ; This routine is called to  allocate  memory  from  the  Process  Allocation
                        001E  ; Region  for those requests which need memory only for the life of an image.
                        001E  ; This routine will expand P0 space to extend the process  allocation  region
                        001E  ; when:
                        001E  ;
                        001E  ;         1.  There is not enough space left in the Process Allocation Region to
                        001E  ;             satisfy the request, or
                        001E  ;
                        001E  ;         2.  The space left in the  Process  Allocation  Region  is  less  than
                        001E  ;             SGN$GW_CTLPAGES - SGN$GW_CTLIMGLIM, and
                        001E  ;
                        001E  ;         3.  The linker option NOP0BUFS has not been selected.
                        001E  ;
                        001E  ;         4.  The routine is called at IPL 0.
                        001E  ;
                        001E  ; INPUTS:
                        001E  ;
                        001E  ;         R1 = SIZE OF BLOCK REQUIRED IN BYTES.
                        001E  ;
                        001E  ; OUTPUTS:
                        001E  ;
                        001E  ;         R0 = LOW BIT CLEAR IF MEMORY IS NOT AVAILABLE.
                        001E  ;
                        001E  ;         R0 = LOW BIT SET IF MEMRY ALLOCATED WITH:
                        001E  ;
                        001E  ;                 R1 = SIZE OF ALLOCATED BLOCK
                        001E  ;                 R2 = ADDRESS OF ALLOCATED BLOCK
                        001E  ;
                        001E  ;         R3 Destroyed.
                        001E  ;
                        001E  ; ENVIRONMENT:
                        001E  ;
                        001E  ;         Called from EXEC or KERNEL modes, this routine will switch
                        001E  ;         into KERNEL mode and disable ASTs to perform the actual
                        001E  ;         allocation.
                        001E  ;
                        001E  ;         IPL = 0 or ASTDEL
                        001E  ;
                        001E  ;--
                        001E
                        001E  EXE$ALOP1IMAG::
         51    0F   C0  001E        ADDL    #MASK,R1                        ; round size to next boundry
         51    0F   CA  0021        BICL    #MASK,R1                        ; truncate size back to multiple
             00C2   30  0024        BSBW    P1SYNCH                         ; get to correct mode, AST level
                        0027
                        0027  ;
                        0027  ; Now in Kernel Mode with ASTs disabled
                        0027  ;
                        0027
                        0027
    00000000'9F     D5  0027        TSTL    @#CTL$GL_PRCALLCNT              ; can we get any P1?
               24   15  002D        BLEQ    P0SPACE                         ; branch if not.
```

```
      53   00000000'9F  DE 002F        MOVAL   @#CTL$GQ_ALLOCREG,R3              ; get list head address
                 0181'  30 0036        BSBW    EXE$ALLOCATE                      ; allocate block
                 17 50  E9 0039        BLBC    R0,POSPACE                        ; on error go try P0 space
  00000000'9F      51   C2 003C        SUBL    R1,@#CTL$GL_PRCALLCNT             ; keep track of space in use
                   01   19 0043        BLSS    10$                               ; get P0 space if not enough
                   05      0045        RSB
                           0046
  00000000'9F      51   C0 0046 10$:   ADDL    R1,@#CTL$GL_PRCALLCNT             ; backup count
              50   52   D0 004D        MOVL    R2,R0                             ; set up for DEALLOCATE
                 0243'  30 0050        BSBW    EXE$DEALLOCATE                    ; return space
                           0053
                           0053 POSPACE:
                   50   D4 0053        CLRL    R0                                ; assume failure
6A 00000000'9F     05   E0 0055        BBS     #IMP$V_NOPOBUFS,@#PIO$GW_IIOIMPA,40$ ; branch if cant expand P0
   53   00000000'9F     DE 005D        MOVAL   @#CTL$GQ_POALLOC,R3               ; get P0 list head
                           0064
                 0153'  30 0064 10$:   BSBW    EXE$ALLOCATE                      ; try to get space
                 5D 50  E8 0067        BLBS    R0,40$                            ; branch if available
                           006A
                           006A ;
                           006A ; SPACE NOT AVAILABLE ON P0 LISTHEAD, EXPAND REGION TO GET IT
                           006A ;
                           006A
                 04 AE  D5 006A        TSTL    4(SP)                             ; can we lower IPL?
                   58   12 006D        BNEQ    40$                               ; branch if not
                           006F        SETIPL  #0                                ; lower IPL
                   51   DD 0072        PUSHL   R1                                ; save request size
      51     51  F7 8F  78 0074        ASHL    #-9,R1,R1                         ; get size of request in pages
                   51   D6 0079        INCL    R1                                ; one more page
              10   51   D1 007B        CMPL    R1,#16                            ; at least 16 pages?
                   03   18 007E        BGEQ    20$                               ; branch if so
              51   10   D0 0080        MOVL    #16,R1                            ; set 16 pages then
              52   7E   7E 0083 20$:   MOVAQ   -(SP),R2                          ; get space for RETADR
                           0086        $EXPREG_S -                               ; expand P0
                           0086                PAGCNT  =       R1, -
                           0086                RETADR  =       (R2), -
                           0086                ACMODE  =       #PSL$C_KERNEL, -
                           0086                REGION  =       #0
                 2B 50  E9 0095        BLBC    R0,30$                            ; get out on error
                           0098        $SETPRT_S -                               ; change page protection
                           0098                INADR   =       (R2), -
                           0098                ACMODE  =       #PSL$C_KERNEL, -
                           0098                PROT    =       #PRT$C_UREW
                 13 50  E9 00AD        BLBC    R0,30$                            ; get out on error
              50   8E   7D 00B0        MOVQ    (SP)+,R0                          ; get address range
              51   50   C2 00B3        SUBL    R0,R1                             ; get size of area
                   51   D6 00B6        INCL    R1
                           00B8        SETIPL  #IPL$_ASTDEL                      ; restore IPL
                 01D8'  30 00BB        BSBW    EXE$DEALLOCATE                    ; put new space on list
              51   8E   D0 00BE        MOVL    (SP)+,R1                          ; restore R1
                   A1   11 00C1        BRB     10$                               ; go back and try request
                           00C3
      5E      0C AE     DE 00C3 30$:   MOVAL   12(SP),SP                         ; pop temp space, saved R1
                   05      00C7 40$:   RSB                                       ; and return
                           00C8
```

```
                              00C8          .SBTTL  EXE$DEAP1 - RETURN MEMORY TO PROCESS ALLOCATION REGION
                              00C8
                              00C8   ;+
                              00C8   ; EXE$DEAP1 - RETURN MEMORY TO PROCESS ALLOCATION REGION
                              00C8   ;
                              00C8   ; This routine is called to return memory allocated via the EXE$ALOP1IMAG
                              00C8   ; or EXE$ALOP1PROC routines.
                              00C8   ;
                              00C8   ; INPUTS:
                              00C8   ;
                              00C8   ;       R0 = ADDRESS OF BLOCK TO BE DEALLOCATED.
                              00C8   ;       R1 = SIZE OF BLOCK IN BYTES
                              00C8   ;
                              00C8   ; OUTPUTS:
                              00C8   ;
                              00C8   ;       R0-R3 Destroyed.
                              00C8   ;
                              00C8   ; ENVIRONMENT:
                              00C8   ;
                              00C8   ;       Called from EXEC or KERNEL modes, this routine will switch
                              00C8   ;       into KERNEL mode and disable ASTs to perform the actual
                              00C8   ;       de-allocation.
                              00C8   ;
                              00C8   ;       IPL = 0 or ASTDEL
                              00C8   ;-
                              00C8
                              00C8   EXE$DEAP1::
                              00C8          ;BSBW   CHECKBLOCK                      ; check deall parameters
                  1F     10   00C8          BSBB    P1SYNCH                         ; get to correct mode, AST l
                              00CA
                              00CA   ;
                              00CA   ; Now in Kernel Mode with ASTs disabled
                              00CA   ;
                              00CA
           10 50  1E     E1   00CA          BBC     #30,R0,10$                      ; branch if P0 address
     53   00000000'9F    DE   00CE          MOVAL   @#CTL$GQ_ALLOCREG,R3            ; get list head address
     00000000'9F  51     C0   00D5          ADDL    R1,@#CTL$GL_PRCALLCNT           ; keep track of space
                  07     11   00DC          BRB     20$
                              00DE
     53   00000000'9F    DE   00DE   10$:   MOVAL   @#CTL$GQ_P0ALLOC,R3             ; get P0 list head
               01AE'     30   00E5   20$:   BSBW    EXE$DEALLOCATE                  ; return space
                         05   00E8          RSB
                              00E9
```

```
                     00E9        .SBTTL  P1 SYNCH ROUTINE
                     00E9
                     00E9  ;+
                     00E9  ;
                     00E9  ; This routine is called to switch into kernel mode if necessary.  It
                     00E9  ; calls the subroutine P1KERN via a $CMKRNL service or a CALLG instruction
                     00E9  ; depending on the current mode.  The P1KERN subroutine performs a JSB
                     00E9  ; back to P1SYNCH's return address to perform the actual allocation
                     00E9  ; routine after disabling ASTs.  Upon return from P1KERN, P1SYNCH
                     00E9  ; pops it's return address and RSBs to it's caller's calller.
                     00E9  ;
                     00E9  ;-
                     00E9
                     00E9
                     00E9  P1SYNCH:
            52  DC   00E9          MOVPSL  R2                          ; get PSL
      17 52 18  E1   00EB          BBC     #PSL$V_CURMOD,R2,10$        ; branch if not in EXEC mode
            03  BB   00EF          PUSHR   #^M<R0,R1>                  ; push input args
         53 5E  D0   00F1          MOVL    SP,R3                       ; get argument list address
                     00F4          $CMKRNL_S -                         ; go into kernel mode
                     00F4                  ROUTIN = 30$, -
                     00F4                  ARGLST = (R3)
            0E  BA   0103          POPR    #^M<R1,R2,R3>               ; get return R1,R2, pop return addr
            05       0105          RSB                                 ; return to caller's caller.
                     0106
         53 8E  D0   0106  10$:    MOVL    (SP)+,R3                    ; get routine address
                     0109
                     0109  20$:    DSBINT  #IPL$_ASTDEL                ; raise IPL
         63 16       010F          JSB     (R3)                        ; go back to routine
                     0111          ENBINT                              ; lower IPL
            05       0114          RSB
                     0115
          000C       0115  30$:    .WORD   ^M<R2,R3>
      50 6C 7D       0117          MOVQ    (AP),R0                     ; get input args
   53 08 AC D0       011A          MOVL    8(AP),R3                    ; get routine address
         E9 10       011E          BSBB    20$                         ; synch and call routine
      6C 51 7D       0120          MOVQ    R1,(AP)                     ; return R1,R2 values
            04       0123          RET
                     0124
                     0124          .END
```

```
$$T1                 = 00000000
BUG$_BADALORQSZ        ********     X   02
BUG$_BADDALRQSZ        ********     X   02
BUG$_BADQHDR           ********     X   02
BUG$_DOUBLDEALO        ********     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$V_NOPOBUFS       = 00000005
IOC$ALLOSPT            ********     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$GW_IIOIMPA          ********   X   03
PR$_IPL               = 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
------------------                         --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                        23
_$255$DUA28:[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$:MEMORYALC/UPDATE=(ENH$:MEMORYALC)+EXECML$/LIB
```

0377 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MDAT
LIS

NLDRIVER
LIS

MDATEND
LIS

OPENMSG
LIS

MTFDT
LIS

LOADMREG
LIS

LOADSUB
LIS

MISCDEF
LIS

MBDRIVER
LIS

MEMORYALC
LIS

MUTEX
LIS

NULLPROC
LIS

LOAVEC
LIS