TSPLAS -- Programmed Logical Ad MACRO V05.05 Wednesday 18-Jan-89 12:12
Table of contents

```
                                        .TITLE  TSPLAS -- Programmed Logical Address Space Support
                                        .ENABLE LC
                                        .DSABLE GBL
                                        .ENABL  AMA
                                ;-------------------------------------------------------------------------------
                                ;  This module contains the routines to implement PLAS support for TSX-Plus.
                                ;
                                ;  Copyright (c) 1983, 1984, 1985.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee  USA
                                ;  All rights reserved
                                ;
                                        .PSECT  TSPLAS
000000                          TSPLAS: .RAD50  /PLS/
000000  062763
                                ;
                                ;-------------------------------------------------------------------------------
                                ;  Global definitions
                                ;
                                        .GLOBL  SEGIN,SEGOUT,IASEGS,EMTPLS,PLSXIT,PLSOFF,MAPPLS,PLSINI
                                ;
                                ;-------------------------------------------------------------------------------
                                ;  Global references
                                ;
                                        .GLOBL  LBASE,KPAR6,RC$INM,RC$FLG,RC$LEN,RC$$SZ,OVRHC
                                        .GLOBL  RC$BAS,Q.BLKN,Q.PAR,Q.COMP,SEGCHN,GETSYQ
                                        .GLOBL  Q.WCNT,SYQIO,CS$ERR,CS$EOF,EM$SSE,OUTBSY,OUTSWP
                                        .GLOBL  INBSY,RC$BLK,RC$USE,KEYRCB,KEYPAR
                                        .GLOBL  LBASE,LNBLKS,VPLAS,$INDRN,LSW5
                                        .GLOBL  EMTBLK,BADEMT,RCBBAS,RCBEND,SETERR,R.GSIZ,TRYMEM
                                        .GLOBL  VSWPFL,LNSBLK,R.GID,R.GSTS,EMTXIT
                                        .GLOBL  LSW6,RS.CRR,TRYPLS,INSFIN,TRYRGN,PRIVC2,P2$CGR
                                        .GLOBL  SR$PX,SR$PAR,SR$PDR,SR$FLG,SR.MOR,WC$MAP,WC.MAP,WC.FST
                                        .GLOBL  MEMXPN,URO,WCBBAS,WC$RCB,WC$$SZ,WCBEND,VALADW
                                        .GLOBL  WC$SIZ,WC$PAR,WC$NPR,WC$LEN,WC$VLO,WC$VHI,WC$TRP
                                        .GLOBL  W.NID,W.NBAS,W.NSIZ,WS.CRW,WS.ELW,WS.UNM,WS.OVR
                                        .GLOBL  W.NSTS,WS.MAP,NUMWCB,W.NOFF,W.NLEN,UPDRO,UPARO
                                        .GLOBL  RPDR,RPAR,SETMAP,WC$OFF,W.NRID,CXTPAG,CORUSR
                                        .GLOBL  RC$PAG,LMEMIN,VCORTM,LMINQ,LQUAN,RS.NEW,RS.NAL
                                        .GLOBL  RC$SFA,LCXPAR,FREMEM,CUPARO,CUPDRO,RC$OFF
                                        .GLOBL  RS.GBL,RS.CGR,RS.PVT,R.NAME,RC$NAM,RC$EXI,RC$GBL
                                        .GLOBL  RC$PVT,SHRRCB,SHRRCN,RC$EXC,RC$OWN,RS.AGE,RC$AEP
                                        .GLOBL  RS.EXI,RC$AGE,SWPFRC,SCPGET,SP$CMD,SP$JOB,SP$DW1
                                        .GLOBL  SA$RGN,S$WFM,QNSPND,CHKABT,RC$CNT
                                        .GLOBL  RS.EGR,RS.UNM,RC$LCG,RC$UNM
                                        .GLOBL  RS.DSP,RC$DSP,IDSFLG,$UDSPC,LSW11
                                        .GLOBL  RDDR,UDDRO,CUDDRO,RDAR,UDARO,CUDARO
                                ;
                                ;-------------------------------------------------------------------------------
                                ;
                                ;  Macro to print an error message when a system crash occurs.
                                ;
                                ;  Arguments:
                                ;   MSG = Name of error message to print.
                                ;   ARG = (Optional) argument value to display with error message.
                                ;
                                        .GLOBL  DIEMSG,DIEARG,SYSHLT
                                        .MACRO  DIE     MSG,ARG
```

```
                                        MOV     MSG,@#DIEMSG
                                        .IF     NB,ARG
                                        MOV     ARG,@#DIEARG
                                        .ENDC
                                        CALL    @#SYSHLT
                                        .ENDM   DIE
                                ;
                                ;  Macro to call a system overlay
                                ;
                                        .MACRO  OCALL   ENTADD
                                        .IF     B,ENTADD
                                        .ERROR  ;OCALL without entry address
                                        .ENDC
                                        CALL    OVRHC
                                        .WORD   ENTADD
                                        .ENDM   OCALL
                                ;
                                ;-------------------------------------------------------------------------------
                                ;  Symbolic equates for error codes
                                ;
000000                          EC0     =       0
000001                          EC1     =       1
000002                          EC2     =       2
000003                          EC3     =       3
000004                          EC4     =       4
000005                          EC5     =       5
000006                          EC6     =       6
000007                          EC7     =       7
000010                          EC10    =       10
000011                          EC11    =       11
000012                          EC12    =       12
000014                          EC14    =       14
000015                          EC15    =       15
000020                          EC20    =       20
000021                          EC21    =       21
                                ;
                                ;-------------------------------------------------------------------------------
                                ;  Local data areas
                                ;
000002  000000                  SCBRCB: .WORD   0       ;Address of region control block
000004  000000                  SCBBAS: .WORD   0       ;Memory address for region inswap/outswap
000006  000000                  SCBLEN: .WORD   0       ;Number of 64-byte blocks to read/write
000010  000000                  SCBBLK: .WORD   0       ;Block number in region swap file
000012  000000                  SSMAP:  .WORD   0       ;Address of base of PLAS swap file free block bit tbl
```

```
                                        .SBTTL  EMTPLS -- Dispatch PLAS EMT's
                                ;-----------------------------------------------------------------------
                                ;  EMTPLS is jumped to from TSEMT when a PLAS EMT is executed.
                                ;  It dispatches control to the appropriate routine based on the
                                ;  EMT sub-function code.
                                ;
000014  113700  000000G         EMTPLS: MOVB    EMTBLK,R0       ;Get sub-function code
000020  020027  000007                  CMP     R0,#MXPLAS      ;Make sure it's not too big?
000024  103402                          BLO     1$              ;Br if ok
000026  000137  000000G                 JMP     BADEMT          ;Invalid sub-function code
000032  006300                  1$:     ASL     R0              ;Convert to word-table index
000034  000170  000040'                 JMP     @PLSJMP(R0)     ;Dispatch to processing routine
                                ;
                                ;  PLAS EMT jump vector
                                ;
000040  000056'                 PLSJMP: .WORD   CRRG            ; 0 .CRRG
000042  001510'                         .WORD   ELRG            ; 1 .ELRG
000044  001702'                         .WORD   CRAW            ; 2 .CRAW
000046  002564'                         .WORD   ELAW            ; 3 .ELAW
000050  002320'                         .WORD   MAP             ; 4 .MAP
000052  002630'                         .WORD   UNMAP           ; 5 .UNMAP
000054  002702'                         .WORD   GMCX            ; 6 .GMCX
                                ;
        000007                  MXPLAS  =       <.-PLSJMP>/2
```

```
                                        .SBTTL  .CRRG  -- Create PLAS region
                                ;-------------------------------------------------------------------
                                ;  The .CRRG EMT is used to create a new PLAS region.
                                ;
000056                          CRRG:
                                ;
                                ;  Set up pointer to user's Region Definition Block (in R4)
                                ;
000056  004737  004632'                 CALL    GETUDB          ;Get addr of user's RDB in R4
                                ;
                                ;  Initialize status flags in user's Region Definition Block (RDB)
                                ;
000062  106564  000000G                 MFPD    R.GSTS(R4)      ;Get status flags from user's RDB
000066  042716  000000C                 BIC     #<RS.CRR!RS.UNM!RS.NAL!RS.NEW>,(SP) ;Clear some status flags
000072  011602                          MOV     (SP),R2         ;Save the status flags
000074  106664  000000G                 MTPD    R.GSTS(R4)      ;Store flags back into RDB
                                ;
                                ;  If this is a request to create a D-space region, see if it is allowed
                                ;
000100  113701  000000G                 MOVB    CORUSR,R1       ;Get job index number
000104  032702  000000G                 BIT     #RS.DSP,R2      ;Is this a D-space request?
000110  001416                          BEQ     1$              ;Skip test if not
000112  105737  000000G                 TSTB    IDSFLG          ;Does machine support separate D-space?
000116  001003                          BNE     2$              ;Br if yes
000120  012700  000020                  MOV     #EC20,R0        ;If not, say machine can't do it
000124  000406                          BR      3$              ;Go abort
000126  032761  000000G 000000G 2$:     BIT     #$UDSPC,LSW11(R1) ;Has job enabled separate I- and D-space?
000134  001004                          BNE     1$              ;OK, D-space PLAS region is allowed
000136  012700  000021                  MOV     #EC21,R0        ;If not, say it hasn't been enabled yet
000142  000137  004764'          3$:     JMP     PLSERR          ;abort request
                                ;
                                ;  Determine if we are creating a private or a shared region
                                ;
000146  032702  000000C          1$:     BIT     #<RS.GBL!RS.CGR>,R2 ;Creating a named region?
000152  001411                          BEQ     PVTRGN          ;Br if not
000154  032761  000000G 000000G         BIT     #$INDRN,LSW5(R1);Is IND running?
000162  001005                          BNE     PVTRGN          ;Force its named regions to be private
000164  032702  000000G                 BIT     #RS.PVT,R2      ;Private named region?
000170  001002                          BNE     PVTRGN          ;Br if yes
000172  000137  000666'                 JMP     SHRRGN          ;Go create shared region
```

```
                                        .SBTTL  PVTRGN -- Create a private region
                                ;-----------------------------------------------------------------------
                                ;  Create a region that is private to the creating job.
                                ;  Note, the region may be either a named ("global") region or an unnamed
                                ;  PLAS region.
                                ;
                                ;  Inputs:
                                ;   R2 = Status flags from user's Region Definition Block.
                                ;   R4 = Pointer to RDB.
                                ;
000176                          PVTRGN:
                                ;
                                ;  See if this is a named or unnamed region.
                                ;
000176  032702  000000C                 BIT     #<RS.GBL!RS.CGR>,R2 ;Creating a named region?
000202  001416                          BEQ     15$             ;Br if not
                                ;
                                ;  This is a named region.
                                ;  Try to find an existing local RCB with the same name.
                                ;
000204  004737  003222'                 CALL    FLRCB           ;Look for local RCB with the same name
000210  103404                          BCS     2$              ;Br if not found
                                ;
                                ;  The requested region already exists.
                                ;
000212  004737  003722'                 CALL    CHKATT          ;Make sure attachment is valid
000216  000137  000424'                 JMP     13$             ;Return info to user and exit
                                ;
                                ;  We want to create a new region.
                                ;  See if that is allowed.
                                ;
000222  032702  000000G         2$:     BIT     #RS.CGR,R2      ;Can we create a new region?
000226  001004                          BNE     15$             ;Br if yes
000230  012700  000012                  MOV     #EC12,R0        ;Return error code 12
000234  000137  004764'                 JMP     PLSERR
                                ;
                                ;  Get a free local RCB and initialize it.
                                ;
000240  004737  003476'         15$:    CALL    GLRCB           ;Get a free local RCB
                                ;
                                ;  The address of a free Region Control Block is in R5.
                                ;  Do some initialization of the RCB.
                                ;
000244  052765  000000C 000000G         BIS     #<RC$PVT!RC$EXC!RC$OFF>,RC$FLG(R5) ;Private region
                                ;
                                ;  See if we can find free memory space for the region.
                                ;
000252  016503  000000G                 MOV     RC$PAG(R5),R3   ;Get # 512-byte pages needed by region
000256  001004                          BNE     18$             ;Br if specified size not zero
000260  012700  000010                  MOV     #EC10,R0        ;Return error code 10
000264  000137  004764'                 JMP     PLSERR
000270  010300                  18$:    MOV     R3,R0
000272  004737  000000G                 CALL    TRYMEM          ;See if we can get free memory for the request
000276  103520                          BCS     4$              ;Br if unable to get a free memory area
                                ;
                                ;  We got a free memory space for the region
                                ;
```

```
                                ;
                                ;  Allocate space in region swap file for this region
                                ;
000316  105737  000000G                TSTB    VSWPFL          ;Is this a non-swapping system?
000322  001421                          BEQ     6$              ;Br if non-swapping system
000324  010300                          MOV     R3,R0           ;Get number of blocks needed
000326  004737  006670'                CALL    SWPGET          ;Try to allocate space in swap file
000332  103010                          BCC     5$              ;Br if allocation was successful
000334  072027  000003                ASH     #3,R0           ;Convert largest swap file space to 64-byte
000340  010037  000000G                MOV     R0,URO          ;Return largest size to user in R0
000344  010502                          MOV     R5,R2           ;Get address of RCB to R2 for SEGFRE
000346  004737  006610'                CALL    SEGFRE          ;Release memory allocated for the region
000352  000541                          BR      14$             ;Return error to user
000354  010265  000000G          5$:     MOV     R2,RC$BLK(R5)   ;Save swap file block number for region
000360  052765  000000G 000000G          BIS     #RC$SFA,RC$FLG(R5) ;Remember that space has been allocated
                                ;
                                ;  Allocation is complete.
                                ;
000366  060361  000000G          6$:     ADD     R3,LNSBLK(R1)   ;Increase # pages allocated to regions
000372  060361  000000G                  ADD     R3,LMEMIN(R1)   ;Increase total number of pages used by job
000376  106564  000000G                  MFPD    R.GSTS(R4)      ;Get RDB status flags
000402  052716  000000G                  BIS     #RS.NAL,(SP)    ;Say this is a new allocation
000406  032716  000000G                  BIT     #RS.GBL,(SP)    ;Is this a named region?
000412  001402                          BEQ     19$             ;Br if not
000414  052716  000000G                  BIS     #RS.NEW,(SP)    ;We created a new named region
000420  106664  000000G          19$:    MTPD    R.GSTS(R4)      ;Store flags back into RDB
                                ;
                                ;  Set some status flags in RCB
                                ;
000424  112765  000001  000000G 13$:     MOVB    #1,RC$CNT(R5)   ;Set attachment count to 1
000432  016500  000000G                  MOV     RC$FLG(R5),R0   ;Get current RCB status flags
000436  052700  000000G                  BIS     #RC$USE,R0      ;Say the RCB is active
000442  106564  000000G                  MFPD    R.GSTS(R4)      ;Get status flags from RDB
000446  012602                          MOV     (SP)+,R2
000450  032702  000000G                  BIT     #RS.AGE,R2      ;Automatic elimination wanted?
000454  001402                          BEQ     7$              ;Br if not
000456  052700  000000G                  BIS     #RC$AEP,R0      ;Set automatic-elimination-pending flag
000462  032702  000000G          7$:     BIT     #RS.EXI,R2      ;Eliminate region on exit?
000466  001402                          BEQ     8$              ;Br if not
000470  052700  000000G                  BIS     #RC$EXI,R0      ;Set flag
000474  032702  000000G          8$:     BIT     #RS.DSP,R2      ;Is this a D-space region?
000500  001402                          BEQ     20$             ;Br if not, I-space is default
000502  052700  000000G                  BIS     #RC$DSP,R0      ;If so, remember for mapping time
000506  010065  000000G          20$:    MOV     R0,RC$FLG(R5)   ;Store new flags back into RCB
                                ;
                                ;  Return information to user's region definition block
                                ;
000512  010546                          MOV     R5,-(SP)        ;Return address of RCB to user
000514  106664  000000G                  MTPD    R.GID(R4)
000520  106564  000000G                  MFPD    R.GSTS(R4)      ;Get old status flags from RDB
000524  052716  000000G                  BIS     #RS.CRR,(SP)    ;Set created-region flag
000530  106664  000000G                  MTPD    R.GSTS(R4)      ;Store back into RDB
000534  000137  000000G                  JMP     EMTXIT          ;Finished with EMT
                                ;
```

```
                                        ;  Cannot get a free memory region large enough for region.
                                        ;  Determine the size of the largest region we will allow to be created.
                                        ;
000540  105737  0000006           4$:       TSTB    VSWPFL          ;Is this a non-swapping system?
000544  001440                              BEQ     12$             ;Br if yes -- we can't consolidate memory
000546  004737  0000006                     CALL    TRYPLS          ;Determine # user pages we could get
000552  010046                              MOV     R0,-(SP)        ;Save free memory space
000554  013700  0000006                     MOV     VPLAS,R0        ;Get # blocks allocated for PLAS swap file
000560  005200                              INC     R0              ;Get more than total possible size
000562  004737  006670'                     CALL    SWPGET          ;See how big largest swap file region is
000566  012602                              MOV     (SP)+,R2        ;Get free memory space size
000570  020002                              CMP     R0,R2           ;Save smaller of free memory or swap region
000572  101401                              BLOS    10$
000574  010200                              MOV     R2,R0
                                        ;
                                        ;  We now know the size of the largest possible region that
                                        ;  we can allocate (Size in terms of 512-blocks is now in R0).
                                        ;  See if it is possible to satisify this request.
                                        ;
000576  020300                      10$:     CMP     R3,R0           ;Is request larger than largest possible?
000600  101022                              BHI     12$             ;Br if yes -- Can't possibly do allocation
                                        ;
                                        ;  It is possible to allocate this region, but we will have to do
                                        ;  some job swapping to collect free memory.
                                        ;  Go ahead and allocate space for the region in the region swap file.
                                        ;
000602  010300                              MOV     R3,R0           ;Get # blocks needed for the region
000604  004737  006670'                     CALL    SWPGET          ;Allocate space in the swap file
000610  103416                              BCS     12$             ;This branch should never be taken
000612  010265  0000006                     MOV     R2,RC$BLK(R5)   ;Save block number for region in swap file
000616  052765  000000C 0000006             BIS     #<RC$USE!RC$SFA>,RC$FLG(R5) ;Region active and swap space
                                        ;
                                        ;  Swap job to expand memory and allocate space for region
                                        ;
000624  066100  0000006                     ADD     LNSBLK(R1),R0   ;Add num blocks already used by plas regions
000630  010061  0000006                     MOV     R0,LNSBLK(R1)   ;This is total space for all regions
000634  066100  0000006                     ADD     LNBLKS(R1),R0   ;Add space used by base portion of job
000640  004737  0000006                     CALL    MEMXPN          ;Do job swapping to expand memory space
000644  000667                              BR      13$             ;Go finish up
                                        ;
                                        ;  We are unable to satisfy the request.
                                        ;  Return largest possible region to user in R0.
                                        ;
000646  072027  000003              12$:     ASH     #3,R0           ;Convert # 512-byte blocks to # 64-byte blocks
000652  010037  0000006                     MOV     R0,@R0          ;Return size of largest region in R0
000656  012700  000007              14$:     MOV     #EC7,R0         ;Return error code # 7
000662  000137  004764'                     JMP     PLSERR
```

```
                                        .SBTTL  SHRRGN -- Create a shared region
                                ;-------------------------------------------------------------------------
                                ;  SHRRGN is called to create a shared memory region.
                                ;
                                ;  Inputs:
                                ;   R2 = Status flags from user's Region Definition Block.
                                ;   R4 = Pointer to RDB.
                                ;
000666                          SHRRGN:
                                ;
                                ;  See if we are authorized to access global regions
                                ;
000666  032737  000000G 000000G         BIT     #P2$CGR,PRIVC2  ;Are we authorized for global regions?
000674  001006                          BNE     23$             ;Br if yes
000676  012700  000007                  MOV     #EC7,R0         ;Return error code 7 if not
000702  005037  000000G                 CLR     UR0             ;Max size of region = 0
000706  000137  004764'                 JMP     PLSERR
                                ;
                                ;  Try to find a local RCB that is connected to a shared region with
                                ;  the same name.
                                ;
000712  004737  003322'         23$:    CALL    FLCRCB          ;Look for local RCB connected to global
000716  103406                          BCS     21$             ;Br if cannot find local RCB
000720  016305  000000G                 MOV     RC$BLK(R3),R5   ;Get pointer to associated global RCB
000724  004737  003722'                 CALL    CHKATT          ;See if we can attach to this region
000730  000137  001424'                 JMP     20$             ;Found local RCB
                                ;
                                ;  We cannot find a local RCB that is attached to the region.
                                ;  See if there is a global RCB for the region.
                                ;
000734  004737  003414'         21$:    CALL    FGRCB           ;Look for global RCB for region
000740  103424                          BCS     16$             ;Br if can't find global RCB either
                                ;
                                ;  We found a global RCB for the region but do not have a local RCB.
                                ;  Create a local RCB for the region.
                                ;
000742  004737  003722'                 CALL    CHKATT          ;See if we can attach to this region
000746  010546                          MOV     R5,-(SP)        ;Save address of global RCB
000750  004737  003476'                 CALL    GLRCB           ;Get a free local RCB (R5 = address)
000754  010503                          MOV     R5,R3           ;Get address of local RCB
000756  012605                          MOV     (SP)+,R5        ;Get address of global RCB
000760  052763  000000C 000000G         BIS     #<RC$GBL!RC$EXC!RC$LCG>,RC$FLG(R3) ;Init local flags
000766  005763  000000G                 TST     RC$LEN(R3)      ;Was specified region size 0?
000772  001165                          BNE     4$              ;Br if not
000774  016563  000000G 000000G         MOV     RC$LEN(R5),RC$LEN(R3) ;Copy length info from global RCB
001002  016563  000000G 000000G         MOV     RC$PAG(R5),RC$PAG(R3)
001010  000556                          BR      4$              ;Go transfer info to local RCB
                                ;
                                ;  The named region does not now exist.
                                ;  See if we want to create a new one.
                                ;
001012  032702  000000G         16$:    BIT     #RS.CGR,R2      ;Can we create a new region?
001016  001004                          BNE     17$             ;Br if yes
001020  012700  000012                  MOV     #EC12,R0        ;Return error code 12
001024  000137  004764'                 JMP     PLSERR
                                ;
                                ;  We want to create a new region.
```

```
                                        ;  Get a free local and global RCB.
                                        ;
001030  004737  003476'          17$:   CALL    GLRCB            ;Get a local RCB (R5 = address)
001034  010503                         MOV     R5,R3            ;Get address of local RCB
001036  004737  003542'                 CALL    GGRCB            ;Get a global RCB (R5 = address)
001042  005765  000000G                TST     RC$LEN(R5)       ;Was specified region size 0?
001046  001004                         BNE     2$               ;Br if not
001050  012700  000010                 MOV     #EC10,R0         ;Return error code 10
001054  000137  004764'                 JMP     PLSERR
                                        ;
                                        ;  R3 = Address of free local RCB.
                                        ;  R5 = Address of free global RCB.
                                        ;  Initialize the RCB's.
                                        ;
001060  052765  000000C 000000G  2$:    BIS     #<RC$GBL!RC$EXC>,RC$FLG(R5) ;Initialize status flags
001066  052763  000000C 000000G         BIS     #<RC$GBL!RC$EXC!RC$LCG>,RC$FLG(R3) ;Init local flags
001074  032702  000000G                BIT     #RS.AGE,R2       ;Is automatic elimination wanted?
001100  001403                         BEQ     24$              ;Br if not
001102  052765  000000G 000000G         BIS     #RC$AEP,RC$FLG(R5) ;Remember automatic elim pending
001110  032702  000000G          24$:   BIT     #RS.DSP,R2       ;Is this to be a D-space region?
001114  001403                         BEQ     5$               ;Br if not
001116  052765  000000G 000000G         BIS     #RC$DSP,RC$FLG(R5) ;Remember to use D-space mapping
                                        ;
                                        ;  Determine how much memory space can possibly be allocated for
                                        ;  a shared region.
                                        ;
001124  113701  000000G          5$:    MOVB    CORUSR,R1        ;Get current job index number
001130  105737  000000G                TSTB    VSWPFL           ;Is this a swapping system?
001134  001006                         BNE     7$               ;Br if yes
                                        ;
                                        ;  Do memory allocation for non-swapping system
                                        ;
001136  016500  000000G                MOV     RC$PAG(R5),R0    ;Get requested # of pages
001142  004737  000000G                CALL    TRYMEM           ;See if a large enough area is available
001146  103550                         BCS     10$              ;Br if not
001150  000412                         BR      13$
                                        ;
                                        ;  Do memory allocation for swapping system
                                        ;
001152  004737  000000G          7$:    CALL    TRYPLS           ;Determine largest possible PLAS region
001156  026500  000000G                CMP     RC$PAG(R5),R0    ;Do we want more than is possible?
001162  101142                         BHI     10$              ;Br if yes
001164  016500  000000G                MOV     RC$PAG(R5),R0    ;Get requested number of pages
001170  004737  000000G                CALL    TRYRGN           ;See if area is available now
001174  103405                         BCS     8$               ;Br if not available now
                                        ;
                                        ;  We were able to allocate the region.
                                        ;  Set up information in the Region Control Block.
                                        ;
001176  072227  000003           13$:   ASH     #3,R2            ;Convert 512-byte page # to 64-byte block #
001202  010265  000000G                MOV     R2,RC$BAS(R5)    ;Remember base of allocated area
001206  000446                         BR      15$
                                        ;
                                        ;  The requested memory space is not currently available.
                                        ;
001210  005702                   8$:    TST     R2               ;Is it possible to allocate that much space?
001212  001526                         BEQ     10$              ;Br if not (R0 contains largest space poss.)
```

```
                                        ;
                                        ;  It is possible to get the requested space but we will have to
                                        ;  do some job swapping to consolidate the free space.
                                        ;
001214  052765  000000G 000000G                BIS     #RC$USE,RC$FLG(R5) ;Remember this RCB is in use
001222  005065  000000G                        CLR     RC$BAS(R5)         ;Say no memory allocated yet
001226                                          OCALL   SWPFRC             ;Outswap any jobs in area we want
001234  010546                                  MOV     R5,-(SP)           ;Save address of RCB
001236                                          OCALL   SCPGET             ;Get a swap command packet
001244  112765  000000G 000000G                MOVB    #SA$RGN,SP$CMD(R5);Set command for swapper
001252  110165  000000G                        MOVB    R1,SP$JOB(R5)      ;Set our job number
001256  011665  000000G                        MOV     (SP),SP$DW1(R5) ;Set address of RCB in SCP
001262  012605                                  MOV     (SP)+,R5           ;Get back address of RCB
                                        ;
                                        ;  Suspend our job until the job swapping is completed
                                        ;
001264  012700  000000G                 14$:    MOV     #S$WFM,R0          ;Waiting-for-memory state
001270  004737  000000G                         CALL    QNSPND             ;Suspend job till swapping completes
001274  004737  000000G                         CALL    CHKABT             ;See if we were aborted while asleep
001300  016500  000000G                         MOV     RC$BAS(R5),R0      ;Was memory allocated for the region?
001304  001767                                  BEQ     14$                ;Br if not
001306  020027  177777                         CMP     R0,#-1             ;Was memory actually allocated by swapper?
001312  001004                                  BNE     15$                ;Br if yes
001314  042765  000000G 000000G                BIC     #RC$USE,RC$FLG(R5) ;Say RCB is not in use
001322  000713                                  BR      7$                 ;Go back and try the whole process again
                                        ;
                                        ;  We have finished allocating space for the region
                                        ;
001324  052765  000000C 000000G 15$:     BIS     #<RC$INM!RC$USE>,RC$FLG(R5);Region in memory & in use
001332  106564  000000G                         MFPD    R.GSTS(R4)         ;Get RDB status flags
001336  052716  000000C                         BIS     #<RS.NEW!RS.NAL>,(SP) ;Say new region was created
001342  106664  000000G                         MTPD    R.GSTS(R4)         ;Store flags back into RDB
                                        ;
                                        ;  Transfer some information from global RCB to local RCB
                                        ;
001346  052763  000000C 000000G 4$:      BIS     #<RC$INM!RC$USE>,RC$FLG(R3) ;Set flags in local RCB too
001354  106564  000000G                         MFPD    R.GSTS(R4)         ;Get flags from user's RDB
001360  032726  000000G                         BIT     #RS.EXI,(SP)+      ;Is elimination on program exit wanted?
001364  001403                                  BEQ     25$                ;Br if not
001366  052763  000000G 000000G                BIS     #RC$EXI,RC$FLG(R3) ;If yes, set automatic elimination flag
001374  032765  000000G 000000G 25$:     BIT     #RC$DSP,RC$FLG(R5) ;Does global region use D-space?
001402  001403                                  BEQ     6$                 ;Br if not
001404  052763  000000G 000000G                BIS     #RC$DSP,RC$FLG(R3) ;If yes, remember to map local thru D-space
001412  016563  000000G 000000G 6$:      MOV     RC$BAS(R5),RC$BAS(R3) ;Base memory page for region
001420  010563  000000G                         MOV     R5,RC$BLK(R3)      ;Set pointer to global RCB in local RCB
                                        ;
                                        ;  Increment attachment counts
                                        ;
001424  105763  000000G                 20$:    TSTB    RC$CNT(R3)         ;Are we already attached to local RCB?
001430  001004                                  BNE     22$                ;Br if yes
001432  005263  000000G                         INC     RC$CNT(R3)         ;Say local attachment count = 1
001436  005265  000000G                         INC     RC$CNT(R5)         ;Increment global attachment count
                                        ;
                                        ;  Return information to user's Region Definition Block
                                        ;
001442  010346                          22$:    MOV     R3,-(SP)           ;Return address of local RCB to user
001444  106664  000000G                         MTPD    R.GID(R4)
```

```
    172 001450  106564  000000G                 MFPD    R.GSTS(R4)      ;Get old status flags from RDB
    173 001454  052716  000000G                 BIS     #RS.CRR,(SP)    ;Set created-region flag
    174 001460  106664  000000G                 MTPD    R.GSTS(R4)      ;Store back into RDB
    175 001464  000137  000000G                 JMP     EMTXIT          ;Finished with EMT
    176                                 ;
    177                                 ;  We cannot allocate as large a region as was requested.
    178                                 ;  Return the size of the largest possible region.
    179                                 ;
    180 001470  072027  000003          10$:    ASH     #3,R0           ;Convert # 512-byte pages to # 64-byte pages
    181 001474  010037  000000G                 MOV     R0,UR0          ;Return largest region size in R0
    182 001500  012700  000007                  MOV     #EC7,R0         ;Return error code 7
    183 001504  000137  004764'                 JMP     PLSERR
```

```
                                        .SBTTL  .ELRG  -- Eliminate a PLAS region
                                ;-----------------------------------------------------------------------
                                ;  .ELRG EMT eliminates a PLAS region.
                                ;
001510  004737  004632'         ELRG:   CALL    GETUDB          ;Get addr of user's Region Definition Blk (R4)
                                ;
                                ;  Get address of Region Control Block
                                ;
001514  106564  000000G                 MFPD    R.GID(R4)       ;Get address of region control block
001520  012602                          MOV     (SP)+,R2
001522  010200                          MOV     R2,R0           ;Get RCB address to R0 for CHKRCB
001524  004737  004714'                 CALL    CHKRCB          ;Make sure RCB address is valid
                                ;
                                ;  Determine if this is a private or shared region.
                                ;
001530  032762  000000G 000000G         BIT     #RC$LCG,RC$FLG(R2) ;Is this a shared region?
001536  001402                          BEQ     ELMPVT          ;Br if not
001540  000137  001576'                 JMP     ELMSHR          ;Eliminate a shared region
```

```
                                        .SBTTL  ELMPVT -- Eliminate a private region
                                ;-----------------------------------------------------------------------
                                ;  Eliminate a private (non-shared) region.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to Region Control Block
                                ;   R4 = Pointer to User's Region Definition Block
                                ;
001544                          ELMPVT:
                                ;
                                ;  Always eliminate unnamed regions
                                ;
001544  032762  000000G 000000G         BIT     #RC$GBL,RC$FLG(R2) ;Is this a named region?
001552  001405                         BEQ     2$                 ;Br if not named
                                ;
                                ;  This is a named region that is being eliminated.
                                ;  Free named region if the RS.EGR flag is set
                                ;
001554  106564  000000G                MFPD    R.GSTS(R4)         ;Get status flags from user's RDB
001560  032726  000000G                BIT     #RS.EGR,(SP)+      ;Eliminate global region flag set?
001564  001426                         BEQ     ELMCOM             ;Br if not -- Just detach from region
                                ;
                                ;  We want to eliminate this region.
                                ;
001566  052762  000000G 000000G 2$:     BIS     #RC$AGE,RC$FLG(R2) ;Force elimination of the region
001574  000422                         BR      ELMCOM             ;Now go detach from region
```

```
                                        .SBTTL  ELMSHR -- Eliminate a shared region
                                ;-------------------------------------------------------------------------
                                ;  Eliminate a shared region.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to local copy of RCB for region.
                                ;   R4 = Pointer to User's Region Definition Block
                                ;
001576  016203  000000G         ELMSHR: MOV     RC$BLK(R2),R3   ;Get pointer to global RCB
                                ;
                                ;  See if we are detaching from the region or eliminating it.
                                ;
001602  106564  000000G                 MFPD    R.GSTS(R4)      ;Get status flags from user's RDB
001606  032726  000000G                 BIT     #RS.EGR,(SP)+   ;Eliminate global region flag set?
001612  001413                          BEQ     ELMCOM          ;Br if not
                                ;
                                ;  We are to eliminate the region.
                                ;  Make sure no one else is attached to region.
                                ;
001614  126327  000000G 000001         CMPB    RC$CNT(R3),#1   ;Are we the only attached job?
001622  001404                          BEQ     1$              ;Br if yes
001624  012700  000014                  MOV     #EC14,R0        ;Return error code 14
001630  000137  004764'                 JMP     PLSERR
001634  052763  000000G 000000G 1$:     BIS     #RC$AGE,RC$FLG(R3) ;Force elimination of global region
                                ;
                                ;  Call RELRGN to do the actual release
                                ;
001642  004737  004010'         ELMCOM: CALL    RELRGN          ;Release the region
                                ;
                                ;  Clean out user's RDB
                                ;
001646  005046                         CLR     -(SP)
001650  106664  000000G                 MTPD    R.GID(R4)       ;Zero Region ID number
001654  005046                         CLR     -(SP)           ;Zero region status word
001656  032762  000000G 000000G        BIT     #RC$UNM,RC$FLG(R2) ;Did we unmap any windows?
001664  001402                          BEQ     9$              ;Br if not
001666  052716  000000G                 BIS     #RS.UNM,(SP)    ;Report to user if we did
001672  106664  000000G         9$:     MTPD    R.GSTS(R4)      ;Return status word
                                ;
                                ;  Finished
                                ;
001676  000137  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  .CRAW  -- Create a virtual address window
                                ;-------------------------------------------------------------------------
                                ;  The .CRAW EMT is used to create a PLAS virtual address window.
                                ;
001702  004737  004632'         CRAW:   CALL    GETUDB          ;Get address of window def block to R4
                                ;
                                ;  Initialize status flags in W.NSTS
                                ;
001706  106564  000000G                 MFPD    W.NSTS(R4)      ;Get current status flags
001712  042716  000000C                 BIC     #WS.ELW!WS.UNM!WS.CRW,(SP) ;Clear misc flags
001716  106664  000000G                 MTPD    W.NSTS(R4)      ;Return flags to user
                                ;
                                ;  Get and check the APR value
                                ;
001722  106514                          MFPD    (R4)            ;Get APR value (high byte)
001724  012602                          MOV     (SP)+,R2
001726  105002                          CLRB    R2              ;Clear low-byte
001730  010200                          MOV     R2,R0           ;Save word with APR number in high-byte
001732  000302                          SWAB    R2              ;Get APR number to low byte
001734  020227  000007                  CMP     R2,#7           ;APR must be in the range 0 to 7
001740  101070                          BHI     2$              ;Br if invalid APR number
                                ;
                                ;  Find a free window control block
                                ;
001742  012705  000000G                 MOV     #WCBBAS,R5      ;Point to 1st WCB
001746  052700  000001                  BIS     #1,R0           ;First WCB is # 1
001752  005765  000000G         1$:     TST     WC$SIZ(R5)      ;Is this WCB in use?
001756  001412                          BEQ     3$              ;Br if not
001760  005200                          INC     R0              ;Keep track of the WCB number
001762  062705  000000G                 ADD     #WC$$SZ,R5      ;Point to next WCB
001766  020527  000000G                 CMP     R5,#WCBEND      ;Have we checked all WCB's?
001772  103767                          BLO     1$              ;Br if not
                                ;
                                ;  Error -- There are no free window control blocks
                                ;
001774  012700  000001                  MOV     #EC1,R0         ;Return error code 1
002000  000137  004764'                 JMP     PLSERR
                                ;
                                ;  We found a free WCB (its address is in R5)
                                ;  Init some cells
                                ;  Return ID number to user
                                ;
002004  112765  177777  000000G 3$:     MOVB    #-1,WC$TRP(R5)  ;Say not associated with any fast maps yet
002012  010046                          MOV     R0,-(SP)        ;Push WCB id number and APR number
002014  106664  000000G                 MTPD    W.NID(R4)       ;Return to user
002020  106564  000000G                 MFPD    W.NOFF(R4)      ;Get specified offset into region
002024  012665  000000G                 MOV     (SP)+,WC$OFF(R5) ;And set into wcb
002030  106564  000000G                 MFPD    W.NRID(R4)      ;Get region identifier
002034  012600                          MOV     (SP)+,R0        ;  from user wdb
002036  004737  004714'                 CALL    CHKRCB          ;Verify that it is valid
002042  010065  000000G                 MOV     R0,WC$RCB(R5)   ;And set into wcb
                                ;
                                ;  Set up information about base virtual address of the window
                                ;
002046  006302                          ASL     R2              ;Get 2*APR number
002050  110265  000000G                 MOVB    R2,WC$PAR(R5)   ;Save APR index
002054  072227  000014                  ASH     #12.,R2         ;Convert APR # to virtual address
```

```
     58 002060  010265  000000G                    MOV     R2,WC$VLO(R5)   ;Save in WCB
     59 002064  010246                             MOV     R2,-(SP)        ;Return virtual address to user
     60 002066  106664  000000G                    MTPD    W.NBAS(R4)
     61                                 ;
     62                                 ;  Set up information about window size
     63                                 ;
     64 002072  106564  000000G                    MFPD    W.NSIZ(R4)      ;Get window size (64-byte units)
     65 002076  012603                             MOV     (SP)+,R3
     66 002100  001410                             BEQ     2$              ;Not legal to have 0 size
     67 002102  000241                             CLC
     68 002104  006002                             ROR     R2              ;Convert base virtual addr to 64-byte units
     69 002106  072227  177773                     ASH     #-5,R2
     70 002112  060302                             ADD     R3,R2           ;Get address above top of region (64-byte u)
     71 002114  020227  002000                     CMP     R2,#2000        ;Is it greater than 64Kb?
     72 002120  101404                             BLOS    4$              ;Br if not too big
     73 002122  012700  000000         2$:         MOV     #ECO,R0         ;Return error code 0
     74 002126  000137  004764'                    JMP     PLSERR
     75 002132  010365  000000G        4$:         MOV     R3,WC$SIZ(R5)   ;Set size of window (64-byte units)
     76 002136  072227  000006                     ASH     #6,R2           ;Get virtual address of top of window
     77 002142  005302                             DEC     R2              ;Get address of last byte in window
     78 002144  010265  000000G                    MOV     R2,WC$VHI(R5)   ;Set high virtual address for window
     79                                 ;
     80                                 ;  See if new window overlaps with any existing windows
     81                                 ;
     82 002150  106564  000000G                    MFPD    W.NSTS(R4)      ;Get user's window status word
     83 002154  012603                             MOV     (SP)+,R3
     84 002156  052703  000000G                    BIS     #WS.CRW,R3      ;Set in successful-creation flag
     85 002162  032703  000000G                    BIT     #WS.OVR,R3      ;Should we permit windows to overlap?
     86 002166  001044                             BNE     7$              ;Skip overlap checking if so
     87                                 ;
     88 002170  012702  000000G                    MOV     #WCBBAS,R2      ;Point to 1st window control block
     89 002174  005762  000000G        5$:         TST     WC$SIZ(R2)      ;Is this window in use?
     90 002200  001432                             BEQ     6$              ;Br if not
     91 002202  020205                             CMP     R2,R5           ;Is this the WCB for the new window?
     92 002204  001430                             BEQ     6$              ;Br if yes
     93 002206  026265  000000G 000000G            CMP     WC$VHI(R2),WC$VLO(R5) ;Is old window below new window?
     94 002214  101424                             BLOS    6$              ;Br if yes
     95 002216  026265  000000G 000000G            CMP     WC$VLO(R2),WC$VHI(R5) ;Is old window above new window?
     96 002224  103020                             BHIS    6$              ;Br if yes
     97                                 ;
     98                                 ;  We have found a window overlap.
     99                                 ;  Eliminate and possibly unmap the old window.
    100                                 ;
    101 002226  052703  000000G                    BIS     #WS.ELW,R3      ;Set flag saying we eliminated a window
    102 002232  005062  000000G                    CLR     WC$SIZ(R2)      ;Eliminate the window
    103 002236  105762  000000G                    TSTB    WC$MAP(R2)      ;Is window mapped now?
    104 002242  001407                             BEQ     58$             ;Br if not
    105 002244  010546                             MOV     R5,-(SP)        ;Save address of WCB for new window
    106 002246  010205                             MOV     R2,R5           ;Get address of WCB of window being unmapped
    107 002250  004737  004500'                    CALL    WCBUMP          ;Unmap the window
    108 002254  012605                             MOV     (SP)+,R5        ;Get back address of WCB of new window
    109 002256  052703  000000G                    BIS     #WS.UNM,R3      ;Say we had to unmap a window
    110 002262  005062  000000G        58$:        CLR     WC$RCB(R2)      ;Dissociate window from region
    111 002266  062702  000000G        6$:         ADD     #WC$$SZ,R2      ;Point to next WCB
    112 002272  020227  000000G                    CMP     R2,#WCBEND      ;Checked all windows?
    113 002276  103736                             BLO     5$              ;Br if more to check
    114 002300  010346                 7$:         MOV     R3,-(SP)        ;Return status flags
```

```
    115 002302  106664  000000G                 MTPD    W.NSTS(R4)      ;Move back to user's area
    116                                 ;
    117                                 ;  If mapping is requested, go do that now
    118                                 ;
    119 002306  032703  000000G                 BIT     #WS.MAP,R3      ;Is mapping of window wanted?
    120 002312  001015                          BNE     MAP2            ;Br if yes
    121 002314  000137  000000G                 JMP     EMTXIT          ;Finished
```

```
                                        .SBTTL  .MAP    -- Map virtual window to physical region
                                ;-----------------------------------------------------------------------
                                ;  The .MAP EMT is used to map a virtual window to a physical region.
                                ;
002320  004737  004632'         MAP:    CALL    GETUDB          ;Get address of window definition block (R4)
                                ;
                                ;  Get window id, check that it is valid and convert it to the address
                                ;  of the window control block.
                                ;
002324  004737  004646'                 CALL    GETWCB          ;Get address of window control block to R5
002330  005765  000000G                 TST     WC$SIZ(R5)      ;Has this window been allocated?
002334  001004                         BNE     MAP2            ;Br if yes
002336  012700  000003                 MOV     #EC3,R0         ;Return error code 3 if window not allocated
002342  000137  004764'                 JMP     PLSERR
                                ;
                                ;  Get address of associated Region Control Block
                                ;
002346  106564  000000G         MAP2:   MFPD    W.NRID(R4)      ;Get address of region control block
002352  012602                         MOV     (SP)+,R2
002354  010200                         MOV     R2,R0           ;Get RCB address to R0 for CHKRCB
002356  004737  004714'                 CALL    CHKRCB          ;Make sure RCB address is ok
                                ;
                                ;  At this point, the following registers are set up:
                                ;    R2 = Address of Region Control Block
                                ;    R4 = Address of Window Definition Block in user's area.
                                ;    R5 = Address of Window Control Block.
                                ;
                                ;  If this window is already mapped set flag saying it is being unmapped.
                                ;
002362  105765  000000G         2$:     TSTB    WC$MAP(R5)      ;Is window currently mapped?
002366  001406                         BEQ     3$              ;Br if not
002370  106564  000000G                 MFPD    W.NSTS(R4)      ;Get current status flags
002374  052716  000000G                 BIS     #WS.UNM,(SP)    ;Set flag saying a window was unmapped
002400  106664  000000G                 MTPD    W.NSTS(R4)      ;Return new flags
                                ;
                                ;  Determine length of window to map
                                ;
002404  106564  000000G         3$:     MFPD    W.NLEN(R4)      ;Get specified mapping length
002410  012603                         MOV     (SP)+,R3
002412  001403                         BEQ     4$              ;If zero, map as much as possible
002414  020365  000000G                 CMP     R3,WC$SIZ(R5)   ;Constrain length to size of window
002420  101402                         BLOS    5$              ;Br if ok
002422  016503  000000G         4$:     MOV     WC$SIZ(R5),R3   ;Try to map full size of window
002426  106564  000000G         5$:     MFPD    W.NOFF(R4)      ;Get window mapping offset
002432  012600                         MOV     (SP)+,R0
002434  010065  000000G                 MOV     R0,WC$OFF(R5)   ;And copy into window control block
002440  020062  000000G                 CMP     R0,RC$LEN(R2)   ;Is offset greater than region length?
002444  103406                         BLO     6$              ;Br if not
002446  004737  004500'                 CALL    WCBUMP          ;Unmap the window
002452  012700  000004                 MOV     #EC4,R0         ;If yes then return error code 4
002456  000137  004764'                 JMP     PLSERR
002462  005400                 6$:     NEG     R0              ;Calculate how much of region is above
002464  066200  000000G                 ADD     RC$LEN(R2),R0   ;  offset
002470  020300                         CMP     R3,R0           ;Constrain mapped length to region size
002472  101401                         BLOS    7$              ;Br if ok
002474  010003                         MOV     R0,R3           ;Get length of region
002476  010365  000000G         7$:     MOV     R3,WC$LEN(R5)   ;Save length of window being mapped
```

```
     58 002502  106564  000000G                    MFPD    W.NLEN(R4)        ;Get user specified length value
     59 002506  005726                             TST     (SP)+             ;Was specified value 0?
     60 002510  001003                             BNE     8$                ;Br if not -- don't change if non-zero
     61 002512  010346                             MOV     R3,-(SP)          ;Return actual length to user
     62 002514  106664  000000G                    MTPD    W.NLEN(R4)
     63 002520  062703  000177          8$:        ADD     #177,R3           ;Determine # PAR's affected by window
     64 002524  072327  177771                     ASH     #-7,R3
     65 002530  120365  000000G                    CMPB    R3,WC$NPR(R5)     ;Are we reducing the # of PAR's mapped?
     66 002534  103002                             BHIS    9$                ;Br if not
     67 002536  004737  004500'                    CALL    WCBUMP            ;Unmap all PAR's before setting new mapping
     68 002542  110365  000000G          9$:        MOVB    R3,WC$NPR(R5)     ;Number of PAR's to change with window
     69 002546  112765  000000G 000000G           MOVB    #WC.MAP,WC$MAP(R5) ;Say window mapped normally (not fastmap)
     70                                     ;
     71                                     ;  Now do the actual mapping
     72                                     ;
     73 002554  004737  004252'                    CALL    WCBMAP            ;Do actual window mapping
     74                                     ;
     75                                     ;  Finished with EMT
     76                                     ;
     77 002560  000137  000000G                    JMP     EMTXIT
```

```
                                        .SBTTL  .ELAW  -- Eliminate a virtual address window
                                ;-------------------------------------------------------------------------
                                ;  The .ELAW EMT eliminates a virtual address window and frees the
                                ;  associated window control block.
                                ;
002564  004737  004632'         ELAW:   CALL    GETUDB          ;Get address of window def block to R4
002570  004737  004646'                 CALL    GETWCB          ;Get address of window control block to R5
                                ;
                                ;  Unmap the window
                                ;
002574  004737  004500'                 CALL    WCBUMP          ;Unmap the window
                                ;
                                ;  Eliminate the window
                                ;
002600  005065  000000G                 CLR     WC$SIZ(R5)      ;Eliminate window and say WCB is free
002604  005065  000000G                 CLR     WC$RCB(R5)      ;And not associated with any region
                                ;
                                ;  Return WS.ELW status flag
                                ;
002610  106564  000000G                 MFPD    W.NSTS(R4)      ;Get current status flags
002614  052716  000000G                 BIS     #WS.ELW,(SP)    ;Set flag
002620  106664  000000G                 MTPD    W.NSTS(R4)      ;Return flags
                                ;
                                ;  Finished
                                ;
002624  000137  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  .UNMAP -- Unmap a virtual window
                                ;-------------------------------------------------------------------
                                ;  The .UNMAP EMT unmaps but does not eliminate a virtual window.
                                ;
002630  004737  004632'         UNMAP:  CALL    GETUDB          ;Get address of window def block to R4
002634  004737  004646'                 CALL    GETWCB          ;Get address of window control block to R5
                                ;
                                ;  Make sure window is currently mapped
                                ;
002640  105765  000000G                 TSTB    WC$MAP(R5)      ;Is window currently mapped?
002644  001004                          BNE     1$              ;Br if yes
002646  012700  000005                  MOV     #EC5,R0         ;Error 5 if not mapped
002652  000137  004764'                 JMP     PLSERR
                                ;
                                ;  Unmap the window
                                ;
002656  004737  004500'         1$:     CALL    WCBUMP          ;Unmap the window
                                ;
                                ;  Set WS.UNM status flag
                                ;
002662  106564  000000G                 MFPD    W.NSTS(R4)      ;Get current status flags
002666  052716  000000G                 BIS     #WS.UNM,(SP)    ;Set flag
002672  106664  000000G                 MTPD    W.NSTS(R4)      ;Return flags
                                ;
                                ;  Finished
                                ;
002676  000137  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  .GMCX  -- Get window mapping status
                                ;-------------------------------------------------------------------
                                ;  The .GMCX EMT is used to get the mapping status of a window.
                                ;  If a window id of 0 is specified, the status of the root portion of the
                                ;  program is returned.
                                ;
002702  004737  004632'         GMCX:   CALL    GETUDB          ;Get address of window def block to R4
                                ;
                                ;  Get window id and see if it is zero
                                ;
002706  106564  000000G                 MFPD    W.NID(R4)       ;Get word with window id
002712  012605                          MOV     (SP)+,R5
002714  042705  177400                  BIC     #177400,R5      ;Clear all but window id
002720  001500                          BEQ     GMCX0           ;Br if window id = 0
                                ;
                                ;  We are getting the status of a PLAS window
                                ;
002722  004737  004646'                 CALL    GETWCB          ;Get address of window control block to R5
002726  005765  000000G                 TST     WC$SIZ(R5)      ;Is this window currenty allocated?
002732  001004                          BNE     1$              ;Br if yes
002734  012700  000003                  MOV     #EC3,R0         ;Return error code 3 if not allocated
002740  000137  004764'                 JMP     PLSERR
                                ;  Set APR number
002744  106564  000000G         1$:     MFPD    W.NID(R4)       ;Get word with APR value and window id
002750  116566  000000G 000001          MOVB    WC$PAR(R5),1(SP);Set APR index in high-order byte
002756  106266  000001                  ASRB    1(SP)           ;Convert index into actual APR number
002762  106664  000000G                 MTPD    W.NID(R4)       ;Return to window def block
                                ;  Set window virtual address
002766  016546  000000G                 MOV     WC$VLO(R5),-(SP);Window virtual address base
002772  106664  000000G                 MTPD    W.NBAS(R4)
                                ;  Set window size
002776  016546  000000G                 MOV     WC$SIZ(R5),-(SP);Window size (64-byte units)
003002  106664  000000G                 MTPD    W.NSIZ(R4)
                                ;  Set region identifier
003006  016500  000000G                 MOV     WC$RCB(R5),R0   ;Get address of region control block
003012  010046                          MOV     R0,-(SP)
003014  106664  000000G                 MTPD    W.NRID(R4)
003020  105765  000000G                 TSTB    WC$MAP(R5)      ;Is window currently mapped?
003024  001420                          BEQ     2$              ;Br if window is not mapped
                                ;
                                ;  Window is mapped to a region.
                                ;  Return offset.
                                ;
003026  016546  000000G                 MOV     WC$OFF(R5),-(SP)
003032  106664  000000G                 MTPD    W.NOFF(R4)
                                ;  Length of mapped portion of window
003036  016546  000000G                 MOV     WC$LEN(R5),-(SP)
003042  106664  000000G                 MTPD    W.NLEN(R4)
                                ;  Status
003046  106564  000000G                 MFPD    W.NSTS(R4)      ;Get current status flags
003052  052716  000000G                 BIS     #WS.MAP,(SP)    ;Say window is mapped
003056  106664  000000G                 MTPD    W.NSTS(R4)
003062  000137  000000G                 JMP     EMTXIT          ;Finished
                                ;
                                ;  Window is not mapped
                                ;
003066  005046                  2$:     CLR     -(SP)
```

```
     58 003070  106664  000000G                 MTPD    W.NOFF(R4)      ;Offset
     59 003074  005046                          CLR     -(SP)
     60 003076  106664  000000G                 MTPD    W.NLEN(R4)      ;Mapped length
     61 003102  106564  000000G                 MFPD    W.NSTS(R4)      ;Get status flags
     62 003106  042716  000000G                 BIC     #WS.MAP,(SP)    ;Clear mapped flag
     63 003112  106664  000000G                 MTPD    W.NSTS(R4)
     64 003116  000137  000000G                 JMP     EMTXIT          ;Finished
     65                                 ;
     66                                 ;  Window id = 0.
     67                                 ;  Return status of root portion of job.
     68                                 ;
     69 003122  005046                  GMCX0:  CLR     -(SP)
     70 003124  106664  000000G                 MTPD    W.NID(R4)       ;Say APR = 0
     71                                 ;  Virtual base address
     72 003130  005046                          CLR     -(SP)           ;Say base address = 0
     73 003132  106664  000000G                 MTPD    W.NBAS(R4)
     74                                 ;  Region size and length
     75 003136  016100  000000G                 MOV     LNBLKS(R1),R0   ;Get # 512-byte blocks assigned to job root
     76 003142  163700  000000G                 SUB     CXTPAG,R0       ;Subtract # blocks used by context block
     77 003146  072027  000003                  ASH     #3,R0           ;Convert to # 64-byte blocks
     78 003152  010046                          MOV     R0,-(SP)
     79 003154  106664  000000G                 MTPD    W.NSIZ(R4)      ;Region size
     80 003160  010046                          MOV     R0,-(SP)
     81 003162  106664  000000G                 MTPD    W.NLEN(R4)      ;Region mapped length
     82                                 ;  Region id
     83 003166  005046                          CLR     -(SP)
     84 003170  106664  000000G                 MTPD    W.NRID(R4)
     85                                 ;  Offset
     86 003174  005046                          CLR     -(SP)
     87 003176  106664  000000G                 MTPD    W.NOFF(R4)
     88                                 ;  Status
     89 003202  106564  000000G                 MFPD    W.NSTS(R4)
     90 003206  052716  000000G                 BIS     #WS.MAP,(SP)
     91 003212  106664  000000G                 MTPD    W.NSTS(R4)
     92 003216  000137  000000G                 JMP     EMTXIT          ;Finished
```

```
                                        .SBTTL  FLRCB  -- Find local RCB for private named region
                                ;-------------------------------------------------------------------------
                                ;  Try to locate a local RCB for a private region whose name matches
                                ;  that specified in the user's RDB.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to user's RDB.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found RCB.
                                ;   C-flag set ==> Could not find RCB.
                                ;   R5 = Pointer to RCB that was found.
                                ;
003222  012705  000000G         FLRCB:  MOV     #RCBBAS,R5      ;Point to first local RCB
                                ;
                                ;  See if this RCB is for a named, private region
                                ;
003226  016500  000000G         1$:     MOV     RC$FLG(R5),R0   ;Get RCB flags
003232  032700  000000G                 BIT     #RC$USE,R0      ;Is this RCB active?
003236  001420                         BEQ     2$              ;Br if not
003240  032700  000000G                 BIT     #RC$GBL,R0      ;Is this for a named region?
003244  001415                         BEQ     2$              ;Br if not
003246  032700  000000G                 BIT     #RC$PVT,R0      ;Is this for a private region?
003252  001412                         BEQ     2$              ;Br if not
                                ;
                                ;  Compare specified name with name in RCB
                                ;
003254  106564  000000G                 MFPD    R.NAME(R4)      ;Get 1st 3 chars of name
003260  022665  000000G                 CMP     (SP)+,RC$NAM(R5);Does first part of name match?
003264  001005                         BNE     2$              ;Br if not
003266  106564  000002G                 MFPD    R.NAME+2(R4)    ;Get 2nd 3 chars of name
003272  022665  000002G                 CMP     (SP)+,RC$NAM+2(R5) ;Do names match?
003276  001407                         BEQ     3$              ;Br if yes -- Found the RCB
                                ;
                                ;  Try next RCB
                                ;
003300  062705  000000G         2$:     ADD     #RC$$SZ,R5      ;Point to next RCB
003304  020527  000000G                 CMP     R5,#RCBEND      ;Checked all RCB's?
003310  103746                         BLO     1$              ;Br if not
                                ;
                                ;  There is no matching RCB
                                ;
003312  000261                         SEC                     ;Signal failure on return
003314  000401                         BR      9$
                                ;
                                ;  We found a matching RCB
                                ;
003316  000241                 3$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
003320  000207                 9$:     RETURN
```

```
                                        .SBTTL  FLCRCB -- Find local RCB for shared region
                                ;-----------------------------------------------------------------------
                                ;  Try to locate a local RCB for a shared region whose name matches
                                ;  that specified in the user's RDB.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to user's RDB.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found RCB.
                                ;   C-flag set ==> Could not find RCB.
                                ;   R3 = Pointer to RCB that was found.
                                ;
003322  012703  000000G       FLCRCB: MOV     #RCBBAS,R3      ;Point to first local RCB
                                ;
                                ;  See if this RCB is for a shared region
                                ;
003326  016300  000000G       1$:     MOV     RC$FLG(R3),R0   ;Get RCB flags
003332  032700  000000G               BIT     #RC$USE,R0      ;Is this RCB active?
003336  001415                       BEQ     2$              ;Br if not
003340  032700  000000G               BIT     #RC$LCG,R0      ;Is this for a shared region?
003344  001412                       BEQ     2$              ;Br if not
                                ;
                                ;  Compare specified name with name in RCB
                                ;
003346  106564  000000G               MFPD    R.NAME(R4)      ;Get 1st 3 chars of name
003352  022663  000000G               CMP     (SP)+,RC$NAM(R3);Does first part of name match?
003356  001005                       BNE     2$              ;Br if not
003360  106564  000002G               MFPD    R.NAME+2(R4)    ;Get 2nd 3 chars of name
003364  022663  000002G               CMP     (SP)+,RC$NAM+2(R3) ;Do names match?
003370  001407                       BEQ     3$              ;Br if yes -- Found the RCB
                                ;
                                ;  Try next RCB
                                ;
003372  062703  000000G       2$:     ADD     #RC$$SZ,R3      ;Point to next RCB
003376  020327  000000G               CMP     R3,#RCBEND      ;Checked all RCB's?
003402  103751                       BLO     1$              ;Br if not
                                ;
                                ;  There is no matching RCB
                                ;
003404  000261                       SEC                     ;Signal failure on return
003406  000401                       BR      9$
                                ;
                                ;  We found a matching RCB
                                ;
003410  000241                3$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
003412  000207                9$:     RETURN
```

```
                                        .SBTTL  FGRCB  -- Find global RCB for shared named region
                                ;-----------------------------------------------------------------------
                                ;  Try to locate a global RCB for a region whose name matches
                                ;  that specified in the user's RDB.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to user's RDB.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found RCB.
                                ;   C-flag set ==> Could not find RCB.
                                ;   R5 = Pointer to RCB that was found.
                                ;
003414  013705  000000G         FGRCB:  MOV     SHRRCB,R5       ;Point to first shared RCB
                                ;
                                ;  See if this RCB is for an active region
                                ;
003420  032765  000000G 000000G 1$:     BIT     #RC$USE,RC$FLG(R5) ;Is this RCB active?
003426  001412                          BEQ     2$              ;Br if not
                                ;
                                ;  Compare specified name with name in RCB
                                ;
003430  106564  000000G                 MFPD    R.NAME(R4)      ;Get 1st 3 chars of name
003434  022665  000000G                 CMP     (SP)+,RC$NAM(R5);Does first part of name match?
003440  001005                          BNE     2$              ;Br if not
003442  106564  000002G                 MFPD    R.NAME+2(R4)    ;Get 2nd 3 chars of name
003446  022665  000002G                 CMP     (SP)+,RC$NAM+2(R5) ;Do names match?
003452  001407                          BEQ     3$              ;Br if yes -- Found the RCB
                                ;
                                ;  Try next RCB
                                ;
003454  062705  000000G         2$:     ADD     #RC$$SZ,R5      ;Point to next RCB
003460  020537  000000G                 CMP     R5,SHRRCN       ;Checked all RCB's?
003464  103755                          BLO     1$              ;Br if not
                                ;
                                ;  There is no matching RCB
                                ;
003466  000261                          SEC                     ;Signal failure on return
003470  000401                          BR      9$
                                ;
                                ;  We found a matching RCB
                                ;
003472  000241                  3$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
003474  000207                  9$:     RETURN
```

```
                                        .SBTTL  GLRCB  -- Get a free local RCB
                                ;-----------------------------------------------------------------------
                                ;  Get a free local RCB and initialize it.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to user's RCB.
                                ;   R2 = Status flags from user's RCB
                                ;
                                ;  Outputs:
                                ;   R5 = Pointer to free RCB.
                                ;
003476  012705  000000G         GLRCB:  MOV     #RCBBAS,R5      ;Point to 1st local RCB
003502  032765  000000G 000000G 1$:     BIT     #RC$USE,RC$FLG(R5) ;Is this RCB free?
003510  001411                          BEQ     2$              ;Br if yes
003512  062705  000000G                 ADD     #RC$$SZ,R5      ;Point to next RCB
003516  020527  000000G                 CMP     R5,#RCBEND      ;Checked all?
003522  103767                          BLO     1$              ;Loop if not
                                ;
                                ;  Error -- No free RCB's
                                ;
003524  012700  000006                  MOV     #EC6,R0         ;Return error code 6
003530  000137  004764'                 JMP     PLSERR
                                ;
                                ;  We found a free RCB.
                                ;  Initialize it
                                ;
003534  004737  003606'         2$:     CALL    INIRCB          ;Initialize the RCB
                                ;
                                ;  Finished
                                ;
003540  000207                          RETURN
```

```
                                        .SBTTL  GGRCB  -- Get a free global RCB
                                ;-------------------------------------------------------------------------
                                ;  Get a free global RCB and initialize it.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to user's RCB.
                                ;   R2 = Status flags from user's RCB
                                ;
                                ;  Outputs:
                                ;   R5 = Pointer to free RCB.
                                ;
003542  013705  000000G         GGRCB:  MOV     SHRRCB,R5       ;Point to 1st shared RCB
003546  032765  000000G 000000G 1$:     BIT     #RC$USE,RC$FLG(R5) ;Is this RCB free?
003554  001411                          BEQ     2$              ;Br if yes
003556  062705  000000G                 ADD     #RC$$SZ,R5      ;Point to next RCB
003562  020537  000000G                 CMP     R5,SHRRCN       ;Checked all?
003566  103767                          BLO     1$              ;Loop if not
                                ;
                                ;  Error -- No free RCB's
                                ;
003570  012700  000006                  MOV     #EC6,R0         ;Return error code 6
003574  000137  004764'                 JMP     PLSERR
                                ;
                                ;  We found a free RCB.
                                ;  Initialize it
                                ;
003600  004737  003606'         2$:     CALL    INIRCB          ;Initialize the RCB
                                ;
                                ;  Finished
                                ;
003604  000207                          RETURN
```

```
                                        .SBTTL  INIRCB -- Initialize a new RCB
                                ;-------------------------------------------------------------------------
                                ;  INIRCB zeroes a new RCB and stores the name of the region into it
                                ;  if it is a named region.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to new RCB to be initialized.
                                ;   R4 = Pointer to user's RDB.
                                ;   R2 = Status flags from RDB.
                                ;
003606  010346                  INIRCB: MOV     R3,-(SP)
                                ;
                                ;  Zero the RCB
                                ;
003610  010503                          MOV     R5,R3           ;Point to the RCB
003612  012700  000000C                 MOV     #RC$$SZ/2,R0    ;Get # words to zero
003616  005023                  1$:     CLR     (R3)+           ;Zero the RCB
003620  077002                          SOB     R0,1$
                                ;
                                ;  Initialize the RCB
                                ;
003622  113765  000000G 000000G         MOVB    CORUSR,RC$OWN(R5) ;Save # of job creating region
                                ;
                                ;  Set up information about the size of the region.
                                ;
003630  106564  000000G                 MFPD    R.GSIZ(R4)      ;Get # 64-byte blocks needed for region
003634  012600                          MOV     (SP)+,R0
003636  010065  000000G                 MOV     R0,RC$LEN(R5)   ;Save region size in RCB
003642  062700  000007                  ADD     #7,R0           ;Bound up to # 512-byte pages
003646  000241                          CLC                     ;Clear carry for rotate
003650  006000                          ROR     R0              ;Convert # 64-byte blocks to # 512-byte blocks
003652  072027  177776                  ASH     #-2,R0
003656  010065  000000G                 MOV     R0,RC$PAG(R5)   ;Save # pages needed for region
                                ;
                                ;  Store the name into the RCB if it is a named region.
                                ;
003662  032702  000000C                 BIT     #<RS.GBL!RS.CGR>,R2 ;Is this a named region?
003666  001413                          BEQ     9$              ;Br if not
003670  052765  000000G 000000G         BIS     #RC$GBL,RC$FLG(R5);Remember this is a named region
003676  106564  000000G                 MFPD    R.NAME(R4)      ;Get 1st 3 chars of name
003702  012665  000000G                 MOV     (SP)+,RC$NAM(R5);Store into RCB
003706  106564  000002G                 MFPD    R.NAME+2(R4)    ;Get 2nd 3 chars of name
003712  012665  000002G                 MOV     (SP)+,RC$NAM+2(R5) ;Store into RCB
                                ;
                                ;  Finished
                                ;
003716  012603                  9$:     MOV     (SP)+,R3
003720  000207                          RETURN
```

```
                                        .SBTTL  CHKATT -- Check valid attachment to named region
                                ;-----------------------------------------------------------------------
                                ;  CHKATT is called when we are about to attach to an existing named
                                ;  region to make sure the size specified in the RDB is compatible with
                                ;  the size of the region.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to user's RDB.
                                ;   R5 = Pointer to RCB for region.
                                ;
003722                          CHKATT:
                                ;
                                ;  See if we are trying to attach to a region that is reserved for
                                ;  exclusive access by another job.
                                ;
003722  032765  000000C 000000G        BIT     #<RC$LCG!RC$PVT>,RC$FLG(R5) ;Private or local copy of global?
003730  001010                         BNE     1$                  ;Br if yes
003732  032765  000000G 000000G        BIT     #RC$EXC,RC$FLG(R5) ;Exclusive access to region reserved?
003740  001404                         BEQ     1$                  ;Br if not
003742  012700  000015                 MOV     #EC15,R0            ;Return error code 15
003746  000137  004764'                JMP     PLSERR
                                ;
                                ;  Get size of region from RDB
                                ;
003752  106564  000000G        1$:     MFPD    R.GSIZ(R4)          ;Get requested size of region
003756  012600                         MOV     (SP)+,R0
003760  001412                         BEQ     9$                  ;Size of zero is ok
003762  020065  000000G                CMP     R0,RC$LEN(R5)       ;Compare with existing length
003766  101407                         BLOS    9$                  ;Br if ok
                                ;
                                ;  Error -- Requested size is too big
                                ;
003770  016537  000000G 000000G        MOV     RC$LEN(R5),UR0      ;Return actual region size to user in R0
003776  012700  000007                 MOV     #EC7,R0             ;Return error code 7
004002  000137  004764'                JMP     PLSERR
                                ;
                                ;  Region size is ok
                                ;
004006  000207                 9$:     RETURN
```

```
                                        .SBTTL  RELRGN -- Release attachment to a region
                                ;-----------------------------------------------------------------------
                                ;  RELRGN is called to release an attachment by the current job to a region.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of Region Control Block for region being released.
                                ;
004010  010246                  RELRGN: MOV     R2,-(SP)
004012  010546                          MOV     R5,-(SP)
                                ;
                                ;  Unmap any windows that are mapped to this region
                                ;
004014  012705  000000G                 MOV     #WCBBAS,R5      ;Point to first window control block
004020  020265  000000G         5$:     CMP     R2,WC$RCB(R5)   ;Is this window associated with this region?
004024  001011                          BNE     6$              ;Br if not
004026  004737  004500'                 CALL    WCBUMP          ;Unmap the window
004032  005065  000000G                 CLR     WC$SIZ(R5)      ;Eliminate it
004036  005065  000000G                 CLR     WC$RCB(R5)      ;And dissociate it from region
004042  052762  000000G 000000G        BIS     #RC$UNM,RC$FLG(R2) ;Remember a window was eliminated
004050  062705  000000G         6$:     ADD     #WC$$SZ,R5      ;Point to next WCB
004054  020527  000000G                 CMP     R5,#WCBEND      ;Checked all windows?
004060  103757                          BLO     5$              ;Br if not
                                ;
                                ;  See if this is a private or shared region.
                                ;
004062  016200  000000G                 MOV     RC$FLG(R2),R0           ;Get RCB status flags
004066  032700  000000G                 BIT     #RC$LCG,R0              ;Private or shared region
004072  001022                          BNE     3$                      ;Br if shared region
                                ;
                                ;  We are releasing a private region
                                ;
004074  032700  000000G                 BIT     #RC$GBL,R0              ;Is this a named region?
004100  001457                          BEQ     1$                      ;Br if not -- Always eliminate
004102  032700  000000G                 BIT     #RC$AGE,R0              ;Should we eliminate the region?
004106  001054                          BNE     1$                      ;Br if yes
004110  105062  000000G                 CLRB    RC$CNT(R2)              ;Say attachment count = 0
004114  032700  000000G                 BIT     #RC$AEP,R0              ;Is automatic elimination pending?
004120  001451                          BEQ     9$                      ;Br if not
004122  042762  000000G 000000G        BIC     #RC$AEP,RC$FLG(R2)      ;Clear pending flag
004130  052762  000000G 000000G        BIS     #RC$AGE,RC$FLG(R2)      ;Set automatic-elimination flag
004136  000442                          BR      9$
                                ;
                                ;  We are releasing a shared region.
                                ;
004140  042762  000000G 000000G 3$:     BIC     #RC$USE,RC$FLG(R2)      ;Say local RCB is free
004146  016202  000000G                 MOV     RC$BLK(R2),R2           ;Get address of global RCB
004152  032700  000000G                 BIT     #RC$AGE,R0              ;Was AGE specified in local RCB?
004156  001403                          BEQ     4$                      ;Br if not
004160  052762  000000G 000000G        BIS     #RC$AGE,RC$FLG(R2)      ;Transfer AGE option to global RCB
004166  016200  000000G         4$:     MOV     RC$FLG(R2),R0           ;Get status flags from global RCB
004172  105362  000000G                 DECB    RC$CNT(R2)              ;Dec attachment count in global RCB
004176  001003                          BNE     2$                      ;Br if someone else still attached
004200  032700  000000G                 BIT     #RC$AGE,R0              ;Is automatic elimination wanted?
004204  001015                          BNE     1$                      ;Br if yes
004206  042762  000000G 000000G 2$:     BIC     #RC$EXC,RC$FLG(R2)      ;Clear exclusive-access flag
004214  032700  000000G                 BIT     #RC$AEP,R0              ;Is automatic elimination pending?
004220  001411                          BEQ     9$                      ;Br if not
```

```
     58 004222  052762  000000G 000000G          BIS     #RC$AGE,RC$FLG(R2)      ;Set automatic elimination flag
     59 004230  042762  000000G 000000G          BIC     #RC$AEP,RC$FLG(R2)      ;Clear automatic elimination pending
     60 004236  000402                          BR      9$
     61                                 ;
     62                                 ;  Eliminate this region
     63                                 ;
     64 004240  004737  006456'         1$:     CALL    FRERCB                  ;Release the region
     65                                 ;
     66                                 ;  Finished
     67                                 ;
     68 004244  012605                  9$:     MOV     (SP)+,R5
     69 004246  012602                          MOV     (SP)+,R2
     70 004250  000207                          RETURN
```

```
                                        .SBTTL  WCBMAP -- Do actual window mapping
                                ;-------------------------------------------------------------------
                                ;  WCBMAP is called to perform the actual mapping of a PLAS window to
                                ;  a physical memory region.  The information in the window control block
                                ;  is used to set up the mapping.  The user PAR and PDR registers are
                                ;  loaded and the RPAR and RPDR cells are set up to reflect the mapping.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of window control block.
                                ;
004252  010146                  WCBMAP: MOV     R1,-(SP)
004254  010246                          MOV     R2,-(SP)
004256  010346                          MOV     R3,-(SP)
004260  010446                          MOV     R4,-(SP)
004262  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get information about region being mapped
                                ;
004264  016502  000000G                 MOV     WC$RCB(R5),R2   ;Get address of region control block
004270  116503  000000G                 MOVB    WC$PAR(R5),R3   ;Get PAR index number (2*base PAR #)
004274  016504  000000G                 MOV     WC$LEN(R5),R4   ;Get # 64-byte blocks to be mapped
004300  016201  000000G                 MOV     RC$BAS(R2),R1   ;Get base 64-byte block # of region
004304  066501  000000G                 ADD     WC$OFF(R5),R1   ;Add window offset within region
                                ;
                                ;  Determine if this window should be mapped thru I- or D-space
                                ;
004310  005005                          CLR     R5              ;Assume I-space mapping
004312  032762  0000000G 000000G       BIT     #RC$DSP,RC$FLG(R2) ;Should this region be in D-space?
004320  001404                          BEQ     4$              ;Br if not
004322  105737  0000000G              TSTB    IDSFLG          ;Does the machine support separate space?
004326  001401                          BEQ     4$              ;Br if not
004330  005205                          INC     R5
                                ;
                                ;  Begin loop to set up each PAR
                                ;
004332  012702  000200          4$:     MOV     #128.,R2        ;Carry 128 in R2
004336  010400                  1$:     MOV     R4,R0           ;Get # 64-byte blocks left to map
004340  001451                          BEQ     2$              ;Br if finished mapping
004342  020002                          CMP     R0,R2           ;We can only map 128 blocks per PAR
004344  101401                          BLOS    3$              ;Br if we can map all that is left
004346  010200                          MOV     R2,R0           ;Map 128 blocks through this PAR
004350  160004                  3$:     SUB     R0,R4           ;Get # 64-byte blocks left after this PAR
004352  005300                          DEC     R0              ;Get # blocks -1
004354  000300                          SWAB    R0              ;Put # blocks in high order byte
004356  052700  000006                  BIS     #6,R0           ;Allow read and write access to page
004362  042700  100261                  BIC     #100261,R0      ;Make sure unused PDR bits are zero
004366  005705                          TST     R5              ;Use I- or D-space registers?
004370  001015                          BNE     5$              ;Br if D-space
004372  010063  0000000G                MOV     R0,UPDR0(R3)    ;Set actual PDR register
004376  010063  0000000G                MOV     R0,RPDR(R3)     ;Set shadow cell
004402  010063  0000000G                MOV     R0,CUPDR0(R3)   ;Set shadow cell in context block
004406  010163  0000000G                MOV     R1,UPAR0(R3)    ;Set actual PAR register
004412  010163  0000000G                MOV     R1,RPAR(R3)     ;Set shadow cell
004416  010163  0000000G                MOV     R1,CUPAR0(R3)   ;Set shadow cell in context block
004422  000414                          BR      6$              ;Step up to next set
004424  010063  0000000G          5$:     MOV     R0,UDDR0(R3)    ;Set actual D-space PDR register
004430  010063  0000000G                  MOV     R0,RDDR(R3)     ;Set shadow cell
```

```
     58 004434  010063  000000G                 MOV     RO,CUDDRO(R3)   ;Set shadow cell in context block
     59 004440  010163  000000G                 MOV     R1,UDARO(R3)    ;Set actual D-space PAR register
     60 004444  010163  000000G                 MOV     R1,RDAR(R3)     ;Set shadow cell
     61 004450  010163  000000G                 MOV     R1,CUDARO(R3)   ;Set shadow cell in context block
     62 004454  060201                  6$:     ADD     R2,R1           ;Advance physical block number
     63 004456  062703  000002                  ADD     #2,R3           ;Advance PAR index
     64 004462  000725                          BR      1$              ;Go load more PAR's
     65                                 ;
     66                                 ;  Finished
     67                                 ;
     68 004464  012605                  2$:     MOV     (SP)+,R5
     69 004466  012604                          MOV     (SP)+,R4
     70 004470  012603                          MOV     (SP)+,R3
     71 004472  012602                          MOV     (SP)+,R2
     72 004474  012601                          MOV     (SP)+,R1
     73 004476  000207                          RETURN
```

```
                                        .SBTTL  WCBUMP -- Do actual unmapping of a PLAS window
                                ;-------------------------------------------------------------------
                                ;  WCBUMP is called to do the actual unmapping operation for a PLAS window.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of window control block
                                ;
004500  010146          WCBUMP: MOV     R1,-(SP)
004502  010246                  MOV     R2,-(SP)
004504  010346                  MOV     R3,-(SP)

004506  105765  000000G         TSTB    WC$MAP(R5)      ;Is the window mapped now?
004512  001443                  BEQ     9$              ;Br if not
004514  016503  000000G         MOV     WC$RCB(R5),R3   ;Get pointer to rcb
004520  116501  000000G         MOVB    WC$PAR(R5),R1   ;Get first APR index number for the window
004524  032763  000000G 000000G BIT     #RC$DSP,RC$FLG(R3) ;Was this mapped thru I- or D-space
004532  001003                  BNE     4$              ;Br if thru D-space
004534  062701  000000G         ADD     #RPDR,R1        ;Index into I-space register set
004540  000402                  BR      5$              ;Step over D-space offset
004542  062701  000000G 4$:     ADD     #RDDR,R1        ;Index into D-space register set
004546  116502  000000G 5$:     MOVB    WC$TRP(R5),R2   ;Get trap region # associated with window
004552  006302                  ASL     R2              ;Convert to index
004554  116500  000000G         MOVB    WC$NPR(R5),R0   ;Get number of APR's affected by window
004560  001416                  BEQ     2$              ;Br if no APR's to unmap
004562  005011          1$:     CLR     (R1)            ;Say no special mapping for this APR
004564  005702                  TST     R2              ;Any trap region to clean up?
004566  100404                  BMI     3$              ;Br if not
004570  005062  000000G         CLR     SR$PDR(R2)      ;Indicate that fast trap region not in use
004574  062702  000002          ADD     #2,R2           ;Advance trap region index
004600  062701  000002  3$:     ADD     #2,R1           ;Advance APR index
004604  077012                  SOB     R0,1$           ;Loop if more APR's to free
                                ;
                                ;  We have reset the mapping information.
                                ;  Call SETMAP to do the actual mapping for the job.
                                ;
004606  113701  000000G         MOVB    CORUSR,R1       ;Get current job index number
004612  004737  000000G         CALL    SETMAP          ;Reload map registers for the job
                                ;
                                ;  Clear WC$MAP to indicate window is unmapped
                                ;
004616  105065  000000G 2$:     CLRB    WC$MAP(R5)      ;Say window is now unmapped
                                ;
                                ;  Finished
                                ;
004622  012603          9$:     MOV     (SP)+,R3
004624  012602                  MOV     (SP)+,R2
004626  012601                  MOV     (SP)+,R1
004630  000207                  RETURN
```

```
                                        .SBTTL  GETUDB -- Get address of user definition block
                                ;-------------------------------------------------------------------
                                ;  GETUDB is called to get the address of a Region Definition Block or a
                                ;  Window Definition Block from a user's EMT argument block.
                                ;  The address is checked to make sure it is valid.
                                ;
                                ;  Outputs:
                                ;   R4 = Address of user's control block.
                                ;
004632  013700  000002G       GETUDB: MOV     EMTBLK+2,R0     ;Get address of control block
004636  010004                        MOV     R0,R4           ;Return in R4
004640  004737  000000G               CALL    VALADW          ;Make sure address is valid
004644  000207                        RETURN

                                ;-------------------------------------------------------------------
                                ;  GETWCB is called to get the address of a Window Control Block
                                ;  based on the value of the window id specified in a user's Window
                                ;  Definition Block.
                                ;  If the window id is not valid an emt error code of 3 is returned.
                                ;
                                ;  Inputs:
                                ;   R4 = Address of user's window definition block.
                                ;
                                ;  Outputs:
                                ;   R5 = Address of window control block.
                                ;
004646  106564  000000G       GETWCB: MFPD    W.NID(R4)       ;Get window id
004652  012600                        MOV     (SP)+,R0
004654  042700  177400                BIC     #177400,R0      ;Clear high-order byte
004660  001411                        BEQ     1$              ;Error if window = 0
004662  020027  000000G               CMP     R0,#NUMWCB      ;Make sure id is not too large
004666  101006                        BHI     1$              ;Br if too big
                                ;
                                ;  Convert window id to address of window control block
                                ;
004670  012705  000000C       2$:     MOV     #WCBBAS-WC$$SZ,R5 ;Point to WCB area
004674  062705  000000G       4$:     ADD     #WC$$SZ,R5      ;Point to correct WCB
004700  077003                        SOB     R0,4$           ;Based on ID number
                                ;
                                ;  Finished
                                ;
004702  000207                        RETURN
                                ;
                                ;  Error -- Invalid window id
                                ;
004704  012700  000003        1$:     MOV     #EC3,R0         ;Return emt error code 3
004710  000137  004764'               JMP     PLSERR
```

```
                                        .SBTTL  CHKRCB -- Check address of Region Control Block
                                ;-----------------------------------------------------------------------
                                ;  CHKRCB is called to verify that the address of a region control
                                ;  block is valid.  If the address is not valid, an emt error code
                                ;  number 2 is returned.
                                ;
                                ;  Inputs:
                                ;   R0 = Region control block address to be checked
                                ;
004714  032700  000001          CHKRCB: BIT     #1,R0           ;Make sure address is even
004720  001015                          BNE     1$              ;Br if odd
004722  020027  000000G                 CMP     R0,#RCBBAS      ;Make sure it is within proper region
004726  103403                          BLO     2$
004730  020027  000000G                 CMP     R0,#RCBEND
004734  103406                          BLO     9$
004736  020037  000000G         2$:     CMP     R0,SHRRCB       ;See if this is a shared RCB
004742  103404                          BLO     1$
004744  020037  000000G                 CMP     R0,SHRRCN
004750  101001                          BHI     1$
                                ;
                                ;  Address is ok
                                ;
004752  000207                  9$:     RETURN
                                ;
                                ;  Error -- Invalid region control block address
                                ;
004754  012700  000002          1$:     MOV     #EC2,R0         ;Return EMT error code 2
004760  000137  004764'                  JMP     PLSERR
```

```
                                        .SBTTL  PLSERR -- Report error on PLAS EMT
                                ;-----------------------------------------------------------------------
                                ;  This routine is jumped to when any error occurs while processing
                                ;  a PLAS EMT.  It returns the error code to the calling program
                                ;  with the C-flag set on return from the EMT.
                                ;
                                ;  Inputs:
                                ;   R0 = EMT error code to be returned
                                ;
004764  000137  000000G         PLSERR: JMP     SETERR          ;Enter EMT error reporting routine
```

```
                                        .SBTTL  SEGOUT -- Outswap job regions
                                ;-----------------------------------------------------------------------
                                ;  SEGOUT is called to write all PLAS regions associated with a
                                ;  particular job to the region swap file.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> No regions need to be outswapped for this job
                                ;   C-flag cleared ==> Beginning to swap regions.
                                ;
004770  010246                  SEGOUT: MOV     R2,-(SP)
                                ;
                                ;  Map PAR6 to job context block
                                ;
004772  016137  000000G 000000G         MOV     LCXPAR(R1),@#KPAR6 ;Map kernel par6 to job context block
                                ;
                                ;  Determine if any regions need to be swapped for this job
                                ;
005000  012702  000000G                 MOV     #RCBBAS,R2      ;Point to first region descriptor
005004  032762  000000G 000000G 1$:     BIT     #RC$USE,RC$FLG(R2) ;Is this RCB active?
005012  001410                          BEQ     3$              ;Br if not
005014  032762  000000G 000000G         BIT     #RC$LCG,RC$FLG(R2);Is this a shared region?
005022  001004                          BNE     3$              ;Br if yes
005024  032762  000000G 000000G         BIT     #RC$INM,RC$FLG(R2);Is this region in memory now?
005032  001007                          BNE     2$              ;Br if yes
005034  062702  000000G         3$:     ADD     #RC$$SZ,R2      ;Point to next region descriptor block
005040  020227  000000G                 CMP     R2,#RCBEND      ;Checked all region descriptors?
005044  103757                          BLO     1$              ;Br if not
                                ;
                                ;  There are no regions that need to be outswapped
                                ;
005046  000261                          SEC                     ;Signal no regions to outswap
005050  000416                          BR      9$
                                ;
                                ;  We found a region that needs to be outswapped.
                                ;  Set up information to describe the transfer.
                                ;
005052  010237  000002'          2$:     MOV     R2,SCBRCB       ;Save address of region control block
005056  016237  000000G 000004'         MOV     RC$BAS(R2),SCBBAS;Base 64-byte block # of region
005064  016237  000000G 000006'         MOV     RC$LEN(R2),SCBLEN;# 64-byte blocks to transfer
005072  016237  000000G 000010'         MOV     RC$BLK(R2),SCBBLK;Block # of start in swap file
                                ;
                                ;  Initiate the transfer
                                ;
005100  004737  005112'                 CALL    OSSGO           ;Start the I/O transfer
005104  000241                          CLC                     ;Signal that swapping is being done
                                ;
                                ;  Finished
                                ;
005106  012602                  9$:     MOV     (SP)+,R2
005110  000207                          RETURN
```

```
                                ;-------------------------------------------------------------------------
                                ;  OSSGO is called to initiate an I/O operation to write a PLAS region
                                ;  (or a portion of a region) to the region swap file.
                                ;
                                ;  Inputs:
                                ;   SCBBAS = Base 64-byte block # of start of region memory region
                                ;   SCBLEN = Number of 64-byte blocks to transfer
                                ;   SCBBLK = Block number in swap file where region is to be written
                                ;
005112  010146                  OSSGO:  MOV     R1,-(SP)
005114  010246                          MOV     R2,-(SP)
                                ;
                                ;  Get a system I/O queue element
                                ;
005116  012701  000000G                 MOV     #SEGCHN,R1      ;Point to system swap file I/O channel
005122  004737  000000G                 CALL    GETSYQ          ;Get I/O queue element (address in R1)
                                ;
                                ;  Set up information in I/O queue element for the transfer
                                ;
005126  063761  000010' 000000G         ADD     SCBBLK,Q.BLKN(R1);Block number within file
005134  013761  000004' 000000G         MOV     SCBBAS,Q.PAR(R1);64-byte memory address
005142  012761  005234' 000000G         MOV     #OSSCMP,Q.COMP(R1);Address of completion routine
005150  013700  000006'                 MOV     SCBLEN,R0       ;Get # 64-byte blocks to transfer
005154  020027  001770                 CMP     R0,#1016.       ;Compare to max we can transfer at one time
005160  101402                         BLOS    1$              ;Br if we can transfer all this time
005162  012700  001770                 MOV     #1016.,R0       ;Truncate amt to transfer
005166  160037  000006'          1$:    SUB     R0,SCBLEN       ;Get # blocks left to transfer
005172  060037  000004'                 ADD     R0,SCBBAS       ;Advance address for next transfer
005176  010002                         MOV     R0,R2
005200  072227  177775                 ASH     #-3,R2          ;Convert # 64-byte blocks to # 512-byte blocks
005204  060237  000010'                 ADD     R2,SCBBLK       ;Advance file block number
005210  072027  000005                 ASH     #5,R0           ;Convert to # words to transfer
005214  005400                         NEG     R0              ;Negative ==> Write operation
005216  010061  000000G                 MOV     R0,Q.WCNT(R1)   ;Set word count for transfer
                                ;
                                ;  Queue the I/O request
                                ;
005222  004737  000000G                 CALL    SYQIO           ;Initiate the I/O operation
                                ;
                                ;  Return
                                ;
005226  012602                          MOV     (SP)+,R2
005230  012601                          MOV     (SP)+,R1
005232  000207                          RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  OSSCMP is the I/O completion routine for PLAS region out swapping.
                                ;
005234  032737  000000C 000000G OSSCMP: BIT     #CS$ERR!CS$EOF,SEGCHN ;Did an I/O error occur?
005242  001405                         BEQ     1$              ;Br if not
005244                                 DIE     #EM$SSE         ;System crash if swapping error
                                ;
                                ;  Map PAR 6 to job context block
                                ;
005256  113701  000000G        1$:     MOVB    OUTBSY,R1       ;Get index number of job being swapped
005262  016137  000000G 000000G       MOV     LCXPAR(R1),@#KPAR6 ;Map kernel par6 to job context block
                                ;
                                ;  See if there is more of this region to transfer
                                ;
005270  005737  000006'                TST     SCBLEN          ;Any more of region left to transfer?
005274  001403                         BEQ     2$              ;Br if not
                                ;
                                ;  Transfer next portion of region
                                ;
005276  004737  005112'                CALL    OSSGO           ;Initiate next transfer
005302  000407                         BR      3$
                                ;
                                ;  Finished swapping this region
                                ;  Indicate that the region is no longer in memory
                                ;
005304  013702  000002'        2$:     MOV     SCBRCB,R2       ;Get address of region control block
005310  004737  006610'                CALL    SEGFRE          ;Free memory space used by the region
                                ;
                                ;  Start outswap of base portion of the job
                                ;
005314                                  OCALL   OUTSWP          ;Start swap of base portion of the job
                                ;
                                ;  Finished
                                ;
005322  000207                 3$:     RETURN
```

```
                                        .SBTTL  IASEGS -- Allocate memory for regions during inswap
                                ;-------------------------------------------------------------------------
                                ;  IASEGS is called to allocate memory regions for PLAS regions for
                                ;  a job that is being inswapped.  The RPAR and RPDR cells are also
                                ;  set up to reflect the current window mapping state.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
005324  010246                  IASEGS: MOV     R2,-(SP)
005326  010346                          MOV     R3,-(SP)
005330  010546                          MOV     R5,-(SP)
                                ;
                                ;  Map PAR6 to the job's context block
                                ;
005332  016137  000000G 000000G         MOV     LCXPAR(R1),@#KPAR6 ;Map kernel par6 to job context block
                                ;
                                ;  Compute address of start of PLAS regions
                                ;
005340  016103  000000G                 MOV     LBASE(R1),R3    ;Base 512-byte block # of job region
005344  066103  000000G                 ADD     LNBLKS(R1),R3   ;Point past job base region
005350  072327  000003                  ASH     #3,R3           ;Convert to 64-byte page number
                                ;
                                ;  Set base address for each region
                                ;
005354  012702  000000G                 MOV     #RCBBAS,R2      ;Point to first region control block
005360  016200  000000G         1$:     MOV     RC$PAG(R2),R0   ;Get # 512-byte pages allocated to region
005364  001416                          BEQ     2$              ;Br if region is not allocated
005366  032762  000000G 000000G         BIT     #RC$LCG,RC$FLG(R2);Is this a shared region?
005374  001012                          BNE     2$              ;Br if yes
005376  072027  000003                  ASH     #3,R0           ;Convert to # 64-byte pages
005402  010362  000000G                 MOV     R3,RC$BAS(R2)   ;Set base 64-byte page # of region
005406  020237  000000G                 CMP     R2,KEYRCB       ;Is this region used for key definitions?
005412  001002                          BNE     3$              ;Br if not
005414  010337  000000G                 MOV     R3,KEYPAR       ;Save pointer to base of key def region
005420  060003                  3$:     ADD     R0,R3           ;Point past end of region's area
005422  062702  000000G         2$:     ADD     #RC$$SZ,R2      ;Point to next region control block
005426  020227  000000G                 CMP     R2,#RCBEND      ;Have we checked all regions?
005432  103752                          BLO     1$              ;Loop if more to allocate
                                ;
                                ;  Set up mapping information for each PLAS window
                                ;
005434  004737  005450'                 CALL    MAPPLS          ;Set up mapping for PLAS regions
                                ;
                                ;  Finished
                                ;
005440  012605                          MOV     (SP)+,R5
005442  012603                          MOV     (SP)+,R3
005444  012602                          MOV     (SP)+,R2
005446  000207                          RETURN
```

```
                                        .SBTTL  MAPPLS -- Set up mapping for all PLAS windows
                                ;-------------------------------------------------------------------------
                                ;  MAPPLS is called to set up the memory mapping for all active PLAS
                                ;  windows for the job.
                                ;
005450  010546                  MAPPLS: MOV     R5,-(SP)
                                ;
                                ;  Begin loop to process each window for the job
                                ;
005452  012705  000000G                 MOV     #WCBBAS,R5      ;Point to first window control block
005456  105765  000000G         3$:     TSTB    WC$MAP(R5)      ;Is this window currently mapped?
005462  001402                          BEQ     4$              ;Br if not
005464  004737  005506'                 CALL    MAPSEG          ;Set up mapping info for this window
005470  062705  000000G         4$:     ADD     #WC$$SZ,R5      ;Point to next window control block
005474  020527  000000G                 CMP     R5,#WCBEND      ;Have we done all windows?
005500  103766                          BLO     3$              ;Br if more to do
                                ;
                                ;  Finished
                                ;
005502  012605                          MOV     (SP)+,R5
005504  000207                          RETURN
```

```
                                        .SBTTL  MAPSEG -- Set up window mapping during an inswap
                                ;-----------------------------------------------------------------------
                                ;  MAPSEG is called to perform the actual mapping of a PLAS window to
                                ;  a physical memory region.  The information in the window control block
                                ;  is used to set up the mapping.
                                ;  The RPAR and RPDR cells are set up to reflect the mapping.
                                ;
                                ;  Inputs:
                                ;   R5 = Address of window control block.
                                ;
                                ;  Outputs:
                                ;   RPAR & RPDR cells set to to reflect correct mapping for windows.
                                ;
005506  010146          MAPSEG: MOV     R1,-(SP)
005510  010246                  MOV     R2,-(SP)
005512  010346                  MOV     R3,-(SP)
005514  010446                  MOV     R4,-(SP)
005516  005046                  CLR     -(SP)           ;Assume mapping will be thru I-space
                                ;
                                ;  Get information about region being mapped
                                ;
005520  016502  000000G         MOV     WC$RCB(R5),R2   ;Get address of region control block
005524  116503  000000G         MOVB    WC$PAR(R5),R3   ;Get PAR index number (2*base PAR #)
005530  016504  000000G         MOV     WC$LEN(R5),R4   ;Get # 64-byte blocks to be mapped
005534  016201  000000G         MOV     RC$BAS(R2),R1   ;Get base 64-byte block # of region
005540  066501  000000G         ADD     WC$OFF(R5),R1   ;Add window offset within region
005544  032762  000000G 000000G BIT     #RC$DSP,RC$FLG(R2) ;Map this window thru D-space?
005552  001401                  BEQ     6$              ;Br if not, use I-space
005554  005216                  INC     (SP)            ;Set flag to use D-space registers
005556  116502  000000G 6$:     MOVB    WC$TRP(R5),R2   ;See if this window has any fast map regions
                                                        ;Relies on sign extension if unused
                                ;
                                ;  Begin loop to set up each PAR
                                ;
005562  010400          1$:     MOV     R4,R0           ;Get # 64-byte blocks left to map
005564  001457                  BEQ     2$              ;Br if finished mapping
005566  020037  000200          CMP     R0,128.         ;We can only map 128 blocks per PAR
005572  101402                  BLOS    3$              ;Br if we can map all that is left
005574  013700  000200          MOV     128.,R0         ;Map 128 blocks through this PAR
005600  160004          3$:     SUB     R0,R4           ;Get # 64-byte blocks left after this PAR
005602  005300                  DEC     R0              ;Get # blocks -1
005604  000300                  SWAB    R0              ;Put # blocks in high order byte
005606  052700  000006          BIS     #6,R0           ;Allow read and write access to page
005612  042700  100261          BIC     #100261,R0      ;Make sure unused PDR bits are zero
005616  005716                  TST     (SP)            ;Are we mapping thru I- or D-space?
005620  001005                  BNE     7$              ;Br if using D-space
005622  010063  000000G         MOV     R0,RPDR(R3)     ;Set shadow cell for PDR
005626  010163  000000G         MOV     R1,RPAR(R3)     ;Set shadow cell for PAR
005632  000404                  BR      8$              ;Skip over D-space cells
005634  010063  000000G 7$:     MOV     R0,RDDR(R3)     ;Set shadow cell for D-space PDR
005640  010163  000000G         MOV     R1,RDAR(R3)     ;Set shadow cell for D-space PAR
                                ;
                                ;  If this window is fast mapped, refresh the fast map cells
                                ;
005644  006302          8$:     ASL     R2              ;Convert trap # to word index
005646  100421                  BMI     5$              ;Skip if no fast map region associated
005650  010062  000000G         MOV     R0,SR$PDR(R2)   ;Set PDR for fast map region
```

```
     58 005654  010162  000000G                    MOV     R1,SR$PAR(R2)    ;Set PAR base for fast map region
     59 005660  006202                             ASR     R2               ;Convert index back to trap #
     60 005662  110362  000000G                    MOVB    R3,SR$PX(R2)     ;Set PAR index
     61 005666  005704                             TST     R4               ;Any more left to map?
     62 005670  001405                             BEQ     4$               ;Br if not
     63 005672  152762  000000G 000000G           BISB    #SR.MOR,SR$FLG(R2) ;There are more cells in this region
     64 005700  005202                             INC     R2               ;Step up to next trap cell for next set
     65 005702  000403                             BR      5$
     66 005704  142762  000000G 000000G 4$:       BICB    #SR.MOR,SR$FLG(R2) ;This is the last cell for this region
     67                                    ;
     68                                    ;  Advance par and par index and repeat until done
     69                                    ;
     70 005712  063701  000200            5$:     ADD     128.,R1          ;Advance physical block number
     71 005716  062703  000002                    ADD     #2,R3            ;Advance PAR index
     72 005722  000717                             BR      1$               ;Go load more PAR's
     73                                    ;
     74                                    ;  Finished
     75                                    ;
     76 005724  005726                     2$:     TST     (SP)+            ;Clean up I- vs. D-space flag on stack
     77 005726  012604                             MOV     (SP)+,R4
     78 005730  012603                             MOV     (SP)+,R3
     79 005732  012602                             MOV     (SP)+,R2
     80 005734  012601                             MOV     (SP)+,R1
     81 005736  000207                             RETURN
```

```
                                        .SBTTL  SEGIN  -- Inswap job regions
                                ;-----------------------------------------------------------------------
                                ;  SEGIN is called to swap into memory any PLAS regions associated with
                                ;  a job.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> No regions need to be brought in
                                ;   C-flag cleared ==> Started inswap of a region
                                ;
005740  010246                  SEGIN:  MOV     R2,-(SP)
                                ;
                                ;  Map PAR6 to the job's context block
                                ;
005742  016137  000000G 000000G         MOV     LCXPAR(R1),@#KPAR6 ;Map kernel par6 to job context block
                                ;
                                ;  Determine if there are any regions that need to be inswapped
                                ;
005750  012702  000000G                 MOV     #RCBBAS,R2      ;Point to first region descriptor block
005754  032762  000000G 000000G 1$:     BIT     #RC$USE,RC$FLG(R2) ;Is this RCB active?
005762  001410                          BEQ     2$              ;Br if not
005764  032762  000000G 000000G         BIT     #RC$LCG,RC$FLG(R2) ;Is this a shared region?
005772  001004                          BNE     2$              ;Br if yes
005774  032762  000000G 000000G         BIT     #RC$INM,RC$FLG(R2);Is this region already in memory?
006002  001407                          BEQ     3$              ;Br if not in memory
006004  062702  000000G         2$:     ADD     #RC$$SZ,R2      ;Point to next region descriptor block
006010  020227  000000G                 CMP     R2,#RCBEND      ;Checked all region descriptors?
006014  103757                          BLO     1$              ;Br if not
                                ;
                                ;  No regions need to be inswapped
                                ;
006016  000261                          SEC                     ;Signal no swapping needed on return
006020  000421                          BR      9$
                                ;
                                ;  We found a region that needs to be inswapped.
                                ;  Set up information to describe the transfer.
                                ;
006022  010237  000002'         3$:     MOV     R2,SCBRCB       ;Remember which region is being swapped
006026  016237  000000G 000004'         MOV     RC$BAS(R2),SCBBAS ;Set base address for the transfer
006034  016237  000000G 000006'         MOV     RC$LEN(R2),SCBLEN ;Set # 64-byte blocks to transfer
006042  016237  000000G 000010'         MOV     RC$BLK(R2),SCBBLK ;Set base block number in swap file
                                ;
                                ;  Set flag saying region is in memory
                                ;
006050  052762  000000G 000000G         BIS     #RC$INM,RC$FLG(R2);Set flag saying region is in memory
                                ;
                                ;  Initiate the transfer
                                ;
006056  004737  006070'                 CALL    ISSGO           ;Start the inswap
006062  000241                          CLC                     ;Signal that we started a swap
                                ;
                                ;  Finished
                                ;
006064  012602                  9$:     MOV     (SP)+,R2
006066  000207                          RETURN
```

```
      1                                ;-------------------------------------------------------------------------------
      2                                ;  ISSGO is called to initiate an inswap of a PLAS region.
      3                                ;
      4                                ;  Inputs:
      5                                ;   SCBBAS = Address in 64-byte units of memory where transfer is to begin
      6                                ;   SCBLEN = Number of 64-byte blocks to transfer
      7                                ;   SCBBLK = Block number in swap file
      8                                ;
      9 006070  010146                 ISSGO:  MOV     R1,-(SP)
     10 006072  010246                         MOV     R2,-(SP)
     11                                ;
     12                                ;  Get a system I/O queue element
     13                                ;
     14 006074  012701  000000G               MOV     #SEGCHN,R1      ;Point to region swapping channel
     15 006100  004737  000000G               CALL    GETSYQ          ;Get a system I/O queue element (R1=address)
     16                                ;
     17                                ;  Set up I/O queue element for the transfer
     18                                ;
     19 006104  063761  000010' 000000G       ADD     SCBBLK,Q.BLKN(R1) ;File block number
     20 006112  013761  000004' 000000G       MOV     SCBBAS,Q.PAR(R1)  ;Base 64-byte memory address
     21 006120  012761  006210' 000000G       MOV     #ISSCMP,Q.COMP(R1);Address of completion routine
     22 006126  013700  000006'               MOV     SCBLEN,R0       ;Get # 64-byte units to transfer
     23 006132  020027  001770                CMP     R0,#1016.       ;Compare to max we can transfer in 1 operation
     24 006136  101402                        BLOS    1$              ;Br if we can read all this time
     25 006140  012700  001770                MOV     #1016.,R0       ;Truncate the transfer
     26 006144  160037  000006'       1$:     SUB     R0,SCBLEN       ;Decrease amt remaining to be transferred
     27 006150  060037  000004'               ADD     R0,SCBBAS       ;Advance base address
     28 006154  010002                        MOV     R0,R2
     29 006156  072227  177775                ASH     #-3,R2          ;Cvt # 64-byte blocks to # 512-byte blocks
     30 006162  060237  000010'               ADD     R2,SCBBLK       ;Advance file block number
     31 006166  072027  000005                ASH     #5,R0           ;Convert to # words to transfer
     32 006172  010061  000000G               MOV     R0,Q.WCNT(R1)   ;Set word count in queue element
     33                                ;
     34                                ;  Queue the I/O request
     35                                ;
     36 006176  004737  000000G               CALL    SYQIO           ;Initiate the I/O operation
     37                                ;
     38                                ;  Finished
     39                                ;
     40 006202  012602                        MOV     (SP)+,R2
     41 006204  012601                        MOV     (SP)+,R1
     42 006206  000207                        RETURN
```

```
                                        ;-------------------------------------------------------------------------
                                        ;  ISSCMP is the I/O completion routine for PLAS region inswaps.
                                        ;
006210  032737  000000C 000000G ISSCMP: BIT     #CS$ERR!CS$EOF,SEGCHN ;Did an I/O error occur?
006216  001405                          BEQ     1$                    ;Br if not
006220                                  DIE     #EM$SSE               ;System crash if region swap error
                                        ;
                                        ;  See if there is more of this region remaining to be transferred
                                        ;
006232  113701  0000000G        1$:     MOVB    INBSY,R1              ;Get index of job being swapped
006236  005737  000006'                  TST     SCBLEN                ;Any portion of region left to transfer?
006242  001403                          BEQ     2$                    ;Br if not
                                        ;
                                        ;  Start transfer of next portion of the region
                                        ;
006244  004737  006070'                  CALL    ISSGO                 ;Start next transfer
006250  000406                          BR      3$
                                        ;
                                        ;  Finished region inswap
                                        ;  See if there are more regions to inswap
                                        ;
006252  004737  005740'         2$:     CALL    SEGIN                 ;Try to start another region inswap
006256  103003                          BCC     3$                    ;Br if started another transfer
                                        ;
                                        ;  Finished inswapping the last region
                                        ;  Say job is completely in memory now
                                        ;
006260                                  OCALL   INSFIN                ;Completed inswap
                                        ;
                                        ;  Finished
                                        ;
006266  000207                  3$:     RETURN
```

```
                                        .SBTTL  PLSXIT -- Do PLAS cleanup on job exit
                                ;-----------------------------------------------------------------------
                                ;  PLSXIT is called from the code that does job cleanup when a job
                                ;  exits or is aborted.
                                ;
006270  010246                  PLSXIT: MOV     R2,-(SP)
006272  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if any regions need to be freed on job exit or abort
                                ;
006274  012702  000000G                 MOV     #RCBBAS,R2      ;Point to first private region control block
006300  016200  000000G         4$:     MOV     RC$FLG(R2),R0   ;Get region status flags
006304  032700  000000G                 BIT     #RC$USE,R0      ;Is this RCB active?
006310  001410                          BEQ     6$              ;Br if not
006312  032700  000000G                 BIT     #RC$EXI,R0      ;Eliminate region on exit?
006316  001403                          BEQ     5$              ;Br if not
006320  052762  000000G 000000G         BIS     #RC$AGE,RC$FLG(R2) ;Force elimination of region
006326  004737  004010'         5$:     CALL    RELRGN          ;Release the region
006332  062702  000000G         6$:     ADD     #RC$$SZ,R2      ;Point to next RCB
006336  020227  000000G                 CMP     R2,#RCBEND      ;Checked all regions?
006342  103756                          BLO     4$              ;Loop if not
                                ;
                                ;  Deallocate any windows
                                ;
006344  012705  000000G                 MOV     #WCBBAS,R5      ;Point to first window control block
006350  004737  004500'         3$:     CALL    WCBUMP          ;Unmap the window
006354  005065  000000G                 CLR     WC$SIZ(R5)      ;Say this window is free
006360  062705  000000G                 ADD     #WC$$SZ,R5      ;Point to next window control block
006364  020527  000000G                 CMP     R5,#WCBEND      ;Done all windows?
006370  103767                          BLO     3$              ;Loop if not
                                ;
                                ;  Finished
                                ;
006372  012605                          MOV     (SP)+,R5
006374  012602                          MOV     (SP)+,R2
006376  000207                          RETURN
```

```
                                        .SBTTL  PLSOFF -- Do PLAS cleanup at job logoff
                                ;-------------------------------------------------------------------
                                ;  PLSOFF is called from the code that does a job logoff to do
                                ;  cleanup processing for PLAS regions.
                                ;
006400  010246                  PLSOFF: MOV     R2,-(SP)
                                ;
                                ;  Detach from all regions
                                ;
006402  012702  000000G                 MOV     #RCBBAS,R2      ;Point to 1st RCB
006406  016200  000000G         1$:     MOV     RC$FLG(R2),R0   ;Get status flags from RCB
006412  032700  000000G                 BIT     #RC$USE,R0      ;Is this RCB active?
006416  001410                          BEQ     2$              ;Br if not
006420  032700  000000C                 BIT     #RC$PVT!RC$OFF,R0 ;Private or release on logoff?
006424  001403                          BEQ     3$              ;Br if not
006426  052762  000000G 000000G         BIS     #RC$AGE,RC$FLG(R2) ;Force release of region
006434  004737  004010'          3$:     CALL    RELRGN          ;Break the attachment
006440  062702  000000G          2$:     ADD     #RC$$SZ,R2      ;Point to next RCB
006444  020227  000000G                 CMP     R2,#RCBEND      ;Checked all RCB's?
006450  103756                          BLO     1$              ;Loop if not
                                ;
                                ;  Finished
                                ;
006452  012602                          MOV     (SP)+,R2
006454  000207                          RETURN
```

```
                        .SBTTL  FRERCB -- Free memory and swap space for a region
                ;-----------------------------------------------------------------------
                ;  FRERCB is called to deallocate memory space and swap file space
                ;  for a plas region.
                ;
                ;  Inputs:
                ;   R2 = Address of Region Control Block
                ;
006456  010546          FRERCB: MOV     R5,-(SP)
                ;
                ;  Return immediately if the region is not allocated
                ;
006460  016200  000000G         MOV     RC$FLG(R2),R0   ;Get status flags from RCB
006464  032700  000000G         BIT     #RC$USE,R0      ;Is this RCB active?
006470  001445                  BEQ     9$              ;Br if not
                ;
                ;  See if this RCB is a local copy of a shared RCB.
                ;
006472  032700  000000G         BIT     #RC$LCG,R0      ;Is this a local copy of shared RCB?
006476  001405                  BEQ     5$              ;Br if not
006500  042762  000000G 000000G BIC     #RC$USE,RC$FLG(R2) ;Say local copy is free
006506  016202  000000G         MOV     RC$BLK(R2),R2   ;Get pointer to shared RCB
                ;
                ;  Free swap file space allocated to the region.
                ;
006512  004737  007132'  5$:     CALL    SWPFRE          ;Free swap file space
                ;
                ;  Free memory space allocated for the region
                ;
006516  004737  006610'         CALL    SEGFRE          ;Free memory space
                ;
                ;  If this is a private region, reduce memory space used by job
                ;
006522  032762  000000G 000000G BIT     #RC$PVT,RC$FLG(R2) ;Is this a private region?
006530  001410                  BEQ     3$              ;Br if not
006532  113705  000000G         MOVB    CORUSR,R5       ;Get job index number
006536  166265  000000G 000000G SUB     RC$PAG(R2),LNSBLK(R5) ;Reduce # pages used by regions for job
006544  166265  000000G 000000G SUB     RC$PAG(R2),LMEMIN(R5) ;Reduce total # pages used by job
                ;
                ;  Mark the region control block as free
                ;
006552  005062  000000G 3$:     CLR     RC$LEN(R2)      ;Say no memory allocated for this region
006556  005062  000000G         CLR     RC$PAG(R2)
006562  005062  000000G         CLR     RC$FLG(R2)
                ;
                ;  If this region was used for key definitions, say it has been deallocated
                ;
006566  020237  000000G         CMP     R2,KEYRCB       ;Was this the key definition region?
006572  001004                  BNE     9$              ;Br if not
006574  005037  000000G         CLR     KEYRCB          ;Say region is gone
006600  005037  000000G         CLR     KEYPAR          ;No user-defined keys
                ;
                ;  Finished
                ;
006604  012605          9$:     MOV     (SP)+,R5
006606  000207                  RETURN
```

```
                                        .SBTTL  SEGFRE -- Free memory space for a region
                                ;-------------------------------------------------------------------
                                ;  SEGFRE is called to deallocate all memory space used by a PLAS region.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of region control block for the region.
                                ;
006610  010146                  SEGFRE: MOV     R1,-(SP)
006612  010246                          MOV     R2,-(SP)
006614  010201                          MOV     R2,R1           ;Carry RCB address in R1
                                ;
                                ;  Free the memory space used by the region
                                ;
006616  016100  000000G                 MOV     RC$LEN(R1),R0   ;Get # 64-byte blocks allocated for region
006622  001412                          BEQ     1$              ;Br if no memory allocated for the region
006624  016102  000000G                 MOV     RC$BAS(R1),R2   ;Get base 64-byte block # for region
006630  000241                          CLC                     ;Convert to 512 byte block number
006632  006002                          ROR     R2
006634  072227  177776                  ASH     #-2,R2
006640  016100  000000G                 MOV     RC$PAG(R1),R0   ;Get # 512-byte pages used by this region
006644  004737  000000G                 CALL    FREMEM          ;Free the memory space used by this region
                                ;
                                ;  Mark region as not in memory
                                ;
006650  042761  000000G 000000G 1$:     BIC     #RC$INM,RC$FLG(R1) ;Say region is not in memory
006656  005061  000000G                 CLR     RC$BAS(R1)
                                ;
                                ;  Finished
                                ;
006662  012602                          MOV     (SP)+,R2
006664  012601                          MOV     (SP)+,R1
006666  000207                          RETURN
```

```
                                        .SBTTL  SWPGET -- Find space in region swap file
                                ;-----------------------------------------------------------------------
                                ;  Try to find space in the region swap file for a PLAS region.
                                ;
                                ;  Inputs:
                                ;   R0 = Number of blocks needed
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Found the requested space.
                                ;   C-flag set     ==> Could not find enough free space.
                                ;   R2 = Starting block number of allocated area.
                                ;   R0 = Size of largest available region if the request could not
                                ;        be satisfied, or the size of the request if the request
                                ;        was satisfied.
                                ;
006670  010146          SWPGET: MOV     R1,-(SP)
006672  010346                  MOV     R3,-(SP)
006674  010446                  MOV     R4,-(SP)
006676  010546                  MOV     R5,-(SP)
006700  010046                  MOV     R0,-(SP)        ;Save requested # blocks on top of stack
                                ;
                                ;  Initialize pointers to bit map
                                ;
006702  013704  000012'         MOV     SSMAP,R4        ;Base of bit map table
006706  005005                  CLR     R5              ;Initialize block number
006710  005000                  CLR     R0              ;Initialize largest region size
                                ;
                                ;  Rapidly skip over allocated blocks at the front of the file
                                ;  Do this by checking 16 blocks at a time
                                ;
006712  013702  000000G         MOV     VPLAS,R2        ;Get total # blocks in swap file
006716  005724          15$:    TST     (R4)+           ;Any free blocks in this word?
006720  001005                  BNE     16$             ;Br if yes
006722  062705  000020          ADD     #16.,R5         ;Advance block number
006726  020502                  CMP     R5,R2           ;Have we gone past end of file?
006730  103772                  BLO     15$             ;Keep looking if not
006732  000470                  BR      12$             ;There are no free blocks
006734  005744          16$:    TST     -(R4)           ;Point back to word with 1st free block
                                ;
                                ;  Search for the start of a free region
                                ;
006736  012701  000001          MOV     #1,R1           ;Get bit mask
006742  013702  000000G 8$:     MOV     VPLAS,R2        ;Get total number of blocks in swap file
006746  020502          1$:     CMP     R5,R2           ;Have we gone past end of the file?
006750  103061                  BHIS    12$             ;Br if yes
006752  030114                  BIT     R1,(R4)         ;Is this block free?
006754  001006                  BNE     2$              ;Br if found a free block
006756  005205                  INC     R5              ;Advance block number
006760  006301                  ASL     R1              ;Shift bit mask for next block
006762  103371                  BCC     1$              ;Br if didn't shift out of end of word
006764  006101                  ROL     R1              ;Rotate bit back into right end of word
006766  005724                  TST     (R4)+           ;Point to next word of bit map
006770  000766                  BR      1$              ;Continue search for free block
                                ;
                                ;  Found the start of a free region in swap file.
                                ;  See if it is big enough to satisfy request.
                                ;
```

```
     58 006772  010502                2$:     MOV     R5,R2           ;Save starting block number
     59 006774  010503                        MOV     R5,R3           ;Calculate block # beyond end of what we need
     60 006776  061603                        ADD     (SP),R3
     61 007000  030114                5$:     BIT     R1,(R4)         ;Is this block free?
     62 007002  001412                        BEQ     7$              ;Br if not free
     63 007004  005205                        INC     R5              ;Advance block number
     64 007006  006301                        ASL     R1              ;Shift bit mask
     65 007010  103002                        BCC     3$              ;Br if did not shift bit out of word
     66 007012  006101                        ROL     R1              ;Shift bit into right end of word
     67 007014  005724                        TST     (R4)+           ;Point to next word in bit map
     68 007016  020503                3$:     CMP     R5,R3           ;Have we found enough space to satisfy request
     69 007020  103011                        BHIS    4$              ;Br if yes
     70 007022  020537  000000G               CMP     R5,VPLAS        ;Have we gone past end of the file?
     71 007026  103764                        BLO     5$              ;Keep searching if not
     72                               ;
     73                               ;  This region was not large enough to satisfy the request.
     74                               ;  Save largest region size.
     75                               ;
     76 007030  010503                7$:     MOV     R5,R3           ;Get number of block beyond end of region
     77 007032  160203                        SUB     R2,R3           ;Calc number of blocks in the region
     78 007034  020300                        CMP     R3,R0           ;Is new region larger than previous one?
     79 007036  101741                        BLOS    8$              ;Br if not
     80 007040  010300                        MOV     R3,R0           ;Save largest region size
     81 007042  000737                        BR      8$
     82                               ;
     83                               ;  We found a region large enough to satisfy the request.
     84                               ;  Claim it.
     85                               ;
     86 007044  010205                4$:     MOV     R2,R5           ;Get starting block number
     87 007046  005004                        CLR     R4              ;Clear high-order for divide
     88 007050  071427  000020               DIV     #16.,R4         ;Convert block # to word and bit index
     89 007054  012701  000001               MOV     #1,R1           ;Get bit mask
     90 007060  072105                        ASH     R5,R1           ;Position bit mask
     91 007062  006304                        ASL     R4              ;Convert word number to byte number
     92 007064  063704  000012'              ADD     SSMAP,R4        ;Point to word within bit map
     93 007070  011603                        MOV     (SP),R3         ;Get request size
     94 007072  040114                10$:    BIC     R1,(R4)         ;Mark blocks as in use
     95 007074  006301                        ASL     R1              ;Shift bit mask
     96 007076  103002                        BCC     9$              ;Br if didn't shift out of word
     97 007100  006101                        ROL     R1              ;Shift bit back into right end of word
     98 007102  005724                        TST     (R4)+           ;Advance to next word of bit map
     99 007104  077306                9$:     SOB     R3,10$          ;Mark all blocks in region as in use
    100 007106  012600                        MOV     (SP)+,R0        ;Return original request size in R0
    101 007110  000241                        CLC                     ;Signal success on return
    102 007112  000402                        BR      13$
    103                              ;
    104                              ;  Could not find a large enough region
    105                              ;
    106 007114  005726                12$:    TST     (SP)+           ;Pop request size off of stack
    107 007116  000261                        SEC                     ;Signal failure on return
    108                              ;
    109                              ;  Finished
    110                              ;
    111 007120  012605                13$:    MOV     (SP)+,R5
    112 007122  012604                        MOV     (SP)+,R4
    113 007124  012603                        MOV     (SP)+,R3
    114 007126  012601                        MOV     (SP)+,R1
```

```
    115 007130  000207                            RETURN
```

```
                                        .SBTTL  SWPFRE -- Free space in region swap file
                                ;-----------------------------------------------------------------------
                                ;  Deallocate space in region swap file.
                                ;
                                ;  Inputs:
                                ;   R2 = Pointer to region descriptor
                                ;
007132  010146                  SWPFRE: MOV     R1,-(SP)
007134  010446                          MOV     R4,-(SP)
007136  010546                          MOV     R5,-(SP)
                                ;
                                ;  Calculate position in bit map of start of region being freed
                                ;
007140  032762  000000G 000000G         BIT     #RC$SFA,RC$FLG(R2) ;Has swap file space been allocated?
007146  001426                          BEQ     9$                 ;Br if not -- nothing to free
007150  016205  000000G                 MOV     RC$BLK(R2),R5      ;Get starting block number
007154  005004                          CLR     R4                 ;Clear high-order for divide
007156  071427  000020                  DIV     #16.,R4            ;Convert block number to word and bit index
007162  012701  000001                  MOV     #1,R1              ;Get a bit mask
007166  072105                          ASH     R5,R1              ;Position bit mask
007170  006304                          ASL     R4                 ;Convert word number to byte number
007172  063704  000012'                 ADD     SSMAP,R4           ;Point to word in bit map
007176  016200  000000G                 MOV     RC$PAG(R2),R0      ;Get # 512-byte pages used by region
                                ;
                                ;  Set bits in bit map saying blocks are free
                                ;
007202  050114                  1$:     BIS     R1,(R4)            ;Set bit saying block is free
007204  006301                          ASL     R1                 ;Shift bit mask
007206  103002                          BCC     2$                 ;Br if didn't shift out of word
007210  006101                          ROL     R1                 ;Shift bit back into right end of word
007212  005724                          TST     (R4)+              ;Point to next word in bit table
007214  077006                  2$:     SOB     R0,1$              ;Set all blocks free
                                ;
                                ;  Say region has no allocated swap file blocks
                                ;
007216  042762  000000G 000000G         BIC     #RC$SFA,RC$FLG(R2) ;Say region has no swap file space
                                ;
                                ;  Finished
                                ;
007224  012605                  9$:     MOV     (SP)+,R5
007226  012604                          MOV     (SP)+,R4
007230  012601                          MOV     (SP)+,R1
007232  000207                          RETURN
```

```
                                        .SBTTL  PLSINI -- Initialize PLAS system
                                ;-------------------------------------------------------------------
                                ;  Initialize the PLAS system during system startup.
                                ;
007234  010246                  PLSINI: MOV     R2,-(SP)
007236  010546                          MOV     R5,-(SP)
                                ;
                                ;  Possibly reduce size of PLAS swap file if we don't have room
                                ;  enough for the required bit map.
                                ;
007240  012705  000000C                 MOV     #137774-PLSTOP,R5 ;Get # bytes of available space in seg
007244  072527  000003                  ASH     #3,R5             ;Get max # blocks we can hold in map
007250  020537  000000G                 CMP     R5,VPLAS          ;Compare with requested amt
007254  103002                          BHIS    2$                ;Br if requested amt ok
007256  010537  000000G                 MOV     R5,VPLAS          ;Reduce size of PLAS swap file
                                ;
                                ;  Allocate a bit map to keep track of free blocks in the swap file
                                ;
007262  012705  007326'         2$:     MOV     #PLSTOP,R5        ;Get pointer past top of PLAS code
007266  010537  000012'                 MOV     R5,SSMAP          ;Set pointer to base of table
007272  013702  000000G                 MOV     VPLAS,R2          ;Get # blocks in PLAS swap file
007276  062702  000017                  ADD     #15.,R2           ;Bound up to word
007302  072227  177774                  ASH     #-4,R2            ;Get # words needed for bit map
007306  042702  170000                  BIC     #170000,R2        ;Clear sign extension
007312  012725  177777          1$:     MOV     #177777,(R5)+     ;Turn all bits on in table
007316  077203                          SOB     R2,1$
                                ;
                                ;  Finished
                                ;
007320  012605                          MOV     (SP)+,R5
007322  012602                          MOV     (SP)+,R2
007324  000207                          RETURN
                                ;
                                ;  Top of PLAS code
                                ;
007326                          PLSTOP:
        000001                          .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  189 Words  ( 1 Pages)
Size of core pool:  18176 Words  ( 71 Pages)
Operating  system:  RT-11

Elapsed time:  00:01:00.11
,LP:TSPLAS=DK:TSPLAS/C/N:SYM
```

Cross reference table (CREF V05.05)

```
$INDRN    1-28      3-36
$UDSPC    1-45      3-27
BADEMT    1-29      2-10
CHKABT    1-43      5-133
CHKATT    4-26      5-25      5-37      20-11#
CHKRCB    6-12      9-50     10-21      25-10#
CORUSR    1-37      3-20      5-84      19-22     23-36     38-36
CRAW      2-18      9-5#
CRRG      2-16      3-5#
CS$EOF    1-26     29-4      35-4
CS$ERR    1-26     29-4      35-4
CUDARO    1-46     22-61*
CUDDRO    1-46     22-58*
CUPARO    1-39     22-54*
CUPDRO    1-39     22-51*
CXTPAG    1-37     13-76
DIEARG    1-56
DIEMSG    1-56     29-6*     35-6*
EC0       1-78#     9-73
EC1       1-79#     9-36
EC10      1-86#     4-50      5-65
EC11      1-87#
EC12      1-88#     4-34      5-54
EC14      1-89#     8-22
EC15      1-90#    20-20
EC2       1-80#    25-27
EC20      1-91#     3-25
EC21      1-92#     3-29
EC3       1-81#    10-13     13-21     24-46
EC4       1-82#    10-50
EC5       1-83#    12-12
EC6       1-84#    17-21     18-21
EC7       1-85#     4-160     5-15      5-182     20-34
ELAW      2-19     11-6#
ELMCOM    7-21      7-26      8-15      8-28#
ELMPVT    6-17      7-9#
ELMSHR    6-18      8-9#
ELRG      2-17      6-5#
EM$SSE    1-26     29-6      35-6
EMTBLK    1-29      2-7      24-10
EMTPLS    1-19      2-7#
EMTXIT    1-30      4-113     5-175     8-42      9-121     10-77     11-26     12-27     13-53     13-64     13-92
FGRCB     5-31     16-14#
FLCRCB    5-22     15-14#
FLRCB     4-21     14-14#
FREMEM    1-39     39-21
FRERCB   21-64     38-9#
GETSYQ    1-25     28-16     34-15
GETUDB    3-9       6-5       9-5      10-5      11-6      12-5      13-7      24-10#
GETWCB   10-10     11-7      12-6      13-18     24-27#
GGRCB     5-62     18-12#
GLRCB     4-39      5-39      5-60     17-12#
GMCX      2-22     13-7#
GMCXO    13-14     13-69#
IASEGS    1-19     30-10#
IDSFLG    1-45      3-23     22-30
```

Cross reference table (CREF V05.05)

```
INBSY     1-27      35-10
INIRCB   17-27      18-27     19-11#
INSFIN    1-31      35-28
ISSCMP   34-21      35-4#
ISSGO    33-51      34-9#     35-16
KEYPAR    1-27      30-35*    38-51*
KEYRCB    1-27      30-33     38-48     38-50*
KPAR6     1-24      27-17*    29-11*    30-16*    33-17*
LBASE     1-24       1-28     30-20
LCXPAR    1-39      27-17     29-11     30-16     33-17
LMEMIN    1-38       4-80*    38-38*
LMINQ     1-38
LNBLKS    1-28       4-151    13-75     30-21
LNSBLK    1-30       4-79*     4-149     4-150*   38-37*
LQUAN     1-38
LSW11     1-45       3-27
LSW5      1-28       3-36
LSW6      1-31
MAP       2-20      10-5#
MAP2      9-120     10-12     10-18#
MAPPLS    1-19      30-43     31-6#
MAPSEG   31-13      32-14#
MEMXPN    1-33       4-152
MXPLAS    2-8        2-24#
NUMWCB    1-36      24-31
OSSCMP   28-22      29-4#
OSSGO    27-47      28-10#    29-20
OUTBSY    1-26      29-10
OUTSWP    1-26      29-31
OVRHC     1-24       5-121     5-123     29-31     35-28
P2$CGR    1-31       5-13
PLSERR    3-30       4-35      4-51      4-161     5-17      5-55      5-66      5-183     8-23      9-37      9-74      10-14
         10-51      12-13     13-22     17-22     18-22     20-21     20-35     24-47     25-28     26-10#
PLSINI    1-19      42-5#
PLSJMP    2-12       2-16#     2-24
PLSOFF    1-19      37-6#
PLSTOP   42-11      42-19     42-36#
PLSXIT    1-19      36-6#
PRIVC2    1-31       5-13
PVTRGN    3-35       3-37      3-39      4-11#
Q.BLKN    1-25      28-20*    34-19*
Q.COMP    1-25      28-22*    34-21*
Q.PAR     1-25      28-21*    34-20*
Q.WCNT    1-26      28-34*    34-32*
QNSPND    1-43       5-132
R.GID     1-30       4-109*    5-171*    6-9       8-33*
R.GSIZ    1-29      19-26     20-25
R.GSTS    1-30       3-13      3-16*     4-81      4-86*     4-93      4-110     4-112*    5-144     5-146*    5-151     5-172
          5-174*     7-19      8-13      8-38*
R.NAME    1-40      14-28     14-31     15-26     15-29     16-23     16-26     19-40     19-42
RC$$SZ    1-24      14-37     15-35     16-32     17-15     18-15     19-16     27-28     30-37     33-28     36-19     37-18
RC$AEP    1-41       4-97      5-76     21-37     21-39     21-56     21-59
RC$AGE    1-42       7-25      8-24     21-34     21-40     21-47     21-49     21-53     21-58     36-17     37-16
RC$BAS    1-25       4-59*     5-108*    5-120*    5-134     5-158     5-158*    22-22     27-41     30-32*    32-25     33-41
         39-16      39-26*
RC$BLK    1-27       4-74*     4-144*    5-24      5-159*    8-9       21-46     27-43     33-43     38-22     41-16
```

Cross reference table (CREF V05.05)

```
RC$CNT    1-43     4-90*    5-163    5-165*   5-166*   8-20     21-36*   21-51*
RC$DSP    1-45     4-103    5-79     5-155    5-157    22-28    23-16    32-27
RC$EXC    1-41     4-44     5-42     5-72     5-73     20-18    21-55
RC$EXI    1-40     4-100    5-154    36-15
RC$FLG    1-24     4-44*    4-60*    4-75*    4-91     4-104*   4-145*   5-42*    5-72*    5-73*    5-76*    5-79*
          5-119*   5-138*   5-143*   5-150*   5-154*   5-155    5-157*   6-16     7-13     7-25*    8-24*    8-35
          14-18    15-18    16-18    17-13    18-13    19-39*   20-16    20-18    21-19*   21-26    21-39*   21-40*
          21-45*   21-49*   21-50    21-55*   21-58*   21-59*   22-28    23-16    27-22    27-24    27-26    30-29
          32-27    33-22    33-24    33-26    33-47*   36-12    36-17*   37-11    37-16*   38-13    38-21*   38-34
          38-44*   39-25*   41-14    41-36*
RC$GBL    1-40     5-42     5-72     5-73     7-13     14-21    19-39    21-32
RC$INM    1-24     4-60     5-143    5-150    27-26    33-26    33-47    39-25
RC$LCG    1-44     5-42     5-73     6-16     15-21    20-16    21-27    27-24    30-29    33-24    38-19
RC$LEN    1-24     5-43     5-45     5-45*    5-63     10-47    10-53    19-28*   20-28    20-33    27-42    33-42
          38-42*   39-14
RC$NAM    1-40     14-29    14-32    15-27    15-30    16-24    16-27    19-41*   19-43*
RC$OFF    1-39     4-44     37-14
RC$OWN    1-41     19-22*
RC$PAG    1-38     4-48     5-46     5-46*    5-90     5-98     5-100    19-33*   30-27    38-37    38-38    38-43*
          39-20    41-23
RC$PVT    1-41     4-44     14-23    20-16    37-14    38-34
RC$SFA    1-39     4-75     4-145    41-14    41-36
RC$UNM    1-44     8-35     21-19
RC$USE    1-27     4-92     4-145    5-119    5-138    5-143    5-150    14-19    15-19    16-18    17-13    18-13
          21-45    27-22    33-22    36-13    37-12    38-14    38-21
RCBBAS    1-29     14-14    15-14    17-12    25-12    27-21    30-26    33-21    36-11    37-10
RCBEND    1-29     14-38    15-36    17-16    25-14    27-29    30-38    33-29    36-20    37-19
RDAR      1-46     22-60*   32-51*
RDDR      1-46     22-57*   23-20    32-50*
RELRGN    8-28     21-8#    36-18    37-17
RPAR      1-37     22-53*   32-48*
RPDR      1-37     22-50*   23-18    32-47*
RS.AGE    1-41     4-95     5-74
RS.CGR    1-40     3-34     4-15     4-32     5-52     19-37
RS.CRR    1-31     3-14     4-111    5-173
RS.DSP    1-45     3-21     4-101    5-77
RS.EGR    1-44     7-20     8-14
RS.EXI    1-42     4-98     5-152
RS.GBL    1-40     3-34     4-15     4-83     19-37
RS.NAL    1-38     3-14     4-82     5-145
RS.NEW    1-38     3-14     4-85     5-145
RS.PVT    1-40     3-38
RS.UNM    1-44     3-14     8-37
S$WFM     1-43     5-131
SA$RGN    1-43     5-124
SCBBAS    1-98#    27-41*   28-21    28-28*   33-41*   34-20    34-27*
SCBBLK    1-100#   27-43*   28-20    28-31*   33-43*   34-19    34-30*
SCBLEN    1-99#    27-42*   28-23    28-27*   29-15    33-42*   34-22    34-26*   35-11
SCBRCB    1-97#    27-40*   29-26    33-40*
SCPGET    1-42     5-123
SEGCHN    1-25     28-15    29-4     34-14    35-4
SEGFRE    4-72     29-27    38-30    39-8#
SEGIN     1-19     33-13#   35-22
SEGOUT    1-19     27-13#
SETERR    1-29     26-10
SETMAP    1-37     23-37
```

Cross reference table (CREF V05.05)

```
SHRRCB    1-41      16-14     18-12     25-16
SHRRCN    1-41      16-33     18-16     25-18
SHRRGN    3-40       5-9#
SP$CMD    1-42       5-124*
SP$DW1    1-42       5-126*
SP$JOB    1-42       5-125*
SR$FLG    1-32      32-63*    32-66*
SR$PAR    1-32      32-58*
SR$PDR    1-32      23-28*    32-57*
SR$PX     1-32      32-60*
SR.MOR    1-32      32-63     32-66
SSMAP     1-101#    40-24     40-92     41-22     42-20*
SWPFRC    1-42       5-121
SWPFRE   38-26      41-8#
SWPGET    4-67       4-124     4-142     40-16#
SYQIO     1-26      28-38     34-36
SYSHLT    1-56      29-6      35-6
TRYMEM    1-29       4-53      5-91
TRYPLS    1-31       4-120     5-97
TRYRGN    1-31       5-101
TSPLAS    1-14#
UDARO     1-46      22-59*
UDDRO     1-46      22-56*
UNMAP     2-21      12-5#
UPARO     1-36      22-52*
UPDRO     1-36      22-49*
URO       1-33       4-70*     4-159*    5-16*     5-181*    20-33*
VALADW    1-33      24-12
VCORTM    1-38
VPLAS     1-28       4-122     40-31     40-43     40-70     42-13     42-15*    42-21
VSWPFL    1-30       4-64      4-118      5-85
W.NBAS    1-35       9-60*     13-30*    13-73*
W.NID     1-35       9-45*     13-11     13-24     13-27*    13-70*    24-27
W.NLEN    1-36      10-38      10-58     10-62*    13-48*    13-60*    13-81*
W.NOFF    1-36       9-46      10-44     13-45*    13-58*    13-87*
W.NRID    1-37       9-48      10-18     13-37*    13-84*
W.NSIZ    1-35       9-64      13-33*    13-79*
W.NSTS    1-36       9-9        9-11*     9-82      9-115*    10-32     10-34*    11-20     11-22*    12-21     12-23*    13-50
         13-52*     13-61      13-63*    13-89     13-91*
WC$$SZ    1-33       9-30       9-111    21-20     24-36     24-37     31-14     36-28
WC$LEN    1-34      10-57*     13-47     22-21     32-24
WC$MAP    1-32       9-103     10-30     10-69*    12-10     13-38     23-12     23-41*    31-11
WC$NPR    1-34      10-65      10-68*    23-23
WC$OFF    1-37       9-47*     10-46*    13-44     22-23     32-26
WC$PAR    1-34       9-56*     13-25     22-20     23-15     32-23
WC$RCB    1-33       9-51*      9-110*   11-16*    13-35     21-14     21-18*    22-19     23-14     32-22
WC$SIZ    1-34       9-27       9-75*     9-89      9-102*    10-11     10-41     10-43     11-15*    13-19     13-32     21-17*
         36-27*
WC$TRP    1-34       9-43*     23-21     32-30
WC$VHI    1-34       9-78*      9-93      9-95
WC$VLO    1-34       9-58*      9-93      9-95     13-29
WC.FST    1-32
WC.MAP    1-32      10-69
WCBBAS    1-33       9-25      9-88      21-13     24-36     31-10     36-25
WCBEND    1-33       9-31      9-112     21-21     31-15     36-29
WCBMAP   10-73      22-11#
```

Cross reference table (CREF V05.05)

WCBUMP	9-107	10-49	10-67	11-11	12-17	21-16	23-8#	36-26
WS.CRW	1-35	9-10	9-84					
WS.ELW	1-35	9-10	9-101	11-21				
WS.MAP	1-36	9-119	13-51	13-62	13-90			
WS.OVR	1-35	9-85						
WS.UNM	1-35	9-10	9-109	10-33	12-22			

Cross reference table (CREF V05.05)

```
DIE       1-57#     29-6      35-6
OCALL     1-67#      5-121     5-123    29-31     35-28
```

Table of contents

```
                                        .TITLE  TSSWAP -- TSX-Plus job swapper
000000                                  .CSECT  TSSWAP
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
000000  075150                  TSSWAP: .RAD50  /SWP/            ;Overlay ID
                                ;
                                ;  TSSWAP contains the routines related to job swapping.
                                ;
                                ;  Copyright (c) 1980,1981,1982,1983,1984,1985.
                                ;  S&H Computer Systems,  Inc.
                                ;  Nashville, Tennessee  USA
                                ;  All rights reserved.
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  TSSWAP
                                        .GLOBL  SWAPER,INSFIN,OUTSWP
                                        .GLOBL  SCPGET
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  OVRHC,INBSY,OUTBSY
                                        .GLOBL  RUNQHD,IOHLTM,VSWPFL,LSW6,FREMEM,Q.COMP,SWPCOT
                                        .GLOBL  LBASE,SWPCHN,LQLINK,GETSYQ,LQUAN,MEMSWP,$INCOR
                                        .GLOBL  LNSBLK,S$CPU,SEGOUT,$MLOCK,INTPRI,EM$SIE
                                        .GLOBL  IASEGS,EM$SFO,LSTSL,LMEMIN,LSW,LSW7,SEGIN,LMINQ
                                        .GLOBL  LIOHLD,$MAPOK,ENQHD,GETMEM,S$$RUN,S$$RT,SCPFHD
                                        .GLOBL  $NDMEM,CS$EOF,Q.WCNT,MAPUSR,LSWPBK,SWPJOB,PSW,LIOCNT
                                        .GLOBL  Q.BLKN,S$IOWT,LNBLKS,RUNQTL,SYQIO,VSWPSL,DOSCHD,Q.PAR
                                        .GLOBL  LSTATE,SWPPOS,CS$ERR,VCORTM,SP$CMD,SP$JOB,EM$LMF
                                        .GLOBL  EM$NSP,SP$LNK,RC$PAG,RC$BAS,TRYRGN,SP$DW1
                                        .GLOBL  FORCEX,$NLOCK,S$WFM
                                ;
                                ;  Macro definitions
                                ;
                                ;  Macro to print an error message when a system crash occurs.
                                ;
                                ;  Arguments:
                                ;   MSG = Name of error message to print.
                                ;   ARG = (Optional) argument value to display with error message.
                                ;
                                        .GLOBL  DIEMSG,DIEARG,SYSHLT
                                        .MACRO  DIE     MSG,ARG
                                        MOV     MSG,@#DIEMSG
                                        .IF     NB,ARG
                                        MOV     ARG,@#DIEARG
                                        .ENDC
                                        CALL    @#SYSHLT
                                        .ENDM   DIE
                                ;
                                ;  Macro definition for call global routines residing in mapped system regions
                                ;
                                        .MACRO  OCALL   ENTADD
                                        .IF     B,ENTADD
                                                .ERROR  ;OCALL SPECIFIED WITH NO ENTRY ADDRESS
                                                .MEXIT
```

```
                                        .ENDC
                                        CALL    OVRHC           ;CALL THE OVERLAY HANDLER
                                        .WORD   ENTADD          ;SPECIFY THE ENTRY POINT
                                        .ENDM
                                ;
                                ;-------------------------------------------------
                                ;  Macros to enable and disable interrupts.
                                ;  DISABL raises the CPU priority level to 7.
                                ;  ENABL  lowers the priority to the current operating
                                ;         priority which is stored in INTPRI.
                                ;
                                        .MACRO  DISABL          ;DISABLE INTERRUPTS
                                        BIS     #340,@#PSW
                                        .ENDM   DISABL
                                ;
                                        .MACRO  ENABL           ;RESTORE INTERRUPT STATUS
                                        BIC     INTPRI,@#PSW
                                        .ENDM   ENABL
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Data areas
                                ;
000002  000000                  SCPQHD: .WORD   0               ;Head of pending swap command packets
000004  000000                  SCPQTL: .WORD   0               ;Tail of pending swap command packets
000006  000000                  SWPSIZ: .WORD   0               ;Number of 512-byte blocks being swapped
000010  000000                  SWPBLK: .WORD   0               ;Block number in swap file where I/O is to go
000012  000000                  SWPBAS: .WORD   0               ;Page number in memory of swap area
                                ;
                                ;  Byte data
                                ;
000014      000                 LSTIN:  .BYTE   0               ;Last job inswapped
000015      000                 LSTOUT: .BYTE   0               ;Last job outswapped
000016      000                 LSTOPR: .BYTE   0               ;Last operation:  0==>Outswap, 1==>Inswap
                                        .EVEN
```

```
                                        .SBTTL  SWAPER -- Job swapper
                                ;-------------------------------------------------------------------------
                                ;  SWAPER is called to see if jobs should be swapped into or out of memory.
                                ;  The basic task of the swapper is simple:  Keep in memory the highest
                                ;  priority jobs that want to run and swap out as few of the lowest priority
                                ;  jobs as necessary to accomplish this.
                                ;
000020                          SWAPER:
                                ;
                                ;  See if we need to outswap any users who are waiting for memory expansion.
                                ;
000020  005737  000000G                 TST     MEMSWP          ;Any jobs need memory-expansion outswap?
000024  001403                          BEQ     12$             ;Br if not
000026  004737  000524'                 CALL    SWPMEM          ;Do memory expansion outswap
000032  103405                          BCS     9$              ;Br if we started a swap
                                ;
                                ;  We have finished doing all forced outswapping.
                                ;  See if there are any special swapper requests pending such as
                                ;  locking jobs in low memory or creating shared PLAS regions.
                                ;
000034  004737  000322'         12$:    CALL    SWPFUN          ;Check for special swapper functions
000040  103402                          BCS     9$              ;Br if special function started
                                ;
                                ;  See if we need to do any job swapping based on job states and priorities.
                                ;
000042  004737  000050'                 CALL    SWPPRI          ;Do normal job swapping
                                ;
                                ;  Finished
                                ;
000046  000207                  9$:     RETURN
```

```
                                        .SBTTL  SWPPRI -- Do job swapping based on states and priorities
                                ;-----------------------------------------------------------------------
                                ;  SWPPRI is the portion of the job swapper that is responsible for doing
                                ;  job swapping to keep in memory the jobs with the highest states and
                                ;  priorities.
                                ;
000050  010146                  SWPPRI: MOV     R1,-(SP)
000052  010246                          MOV     R2,-(SP)
                                ;
                                ;  Scan down the job list from highest priority to lowest priority
                                ;  looking for a job that is ready to run but is not in memory.
                                ;
000054                                  DISABL                  ;** Disable **
000062  113701  000000G                 MOVB    RUNQHD,R1       ;Get head of job list
000066  001507                          BEQ     8$              ;Br if no jobs in list
000070  026127  000000G 000000G 2$:     CMP     LSTATE(R1),#S$$RUN ;Does this job want to run?
000076  101103                          BHI     8$              ;Br if there are no ready-to-run jobs
000100  032761  000000G 000000G         BIT     #$INCOR,LSW(R1) ;Is this job in memory now?
000106  001404                          BEQ     1$              ;Br if not -- go try to get it in
000110  116101  000000G                 MOVB    LQLINK(R1),R1   ;Advance to next job in list
000114  001365                          BNE     2$              ;Br if more to check
000116  000473                          BR      8$              ;No more jobs in list
                                ;
                                ;  We found a job that is ready to run but is not now in memory.
                                ;
                                ;  See if there is enough free memory available to bring in the job.
                                ;
000120                          1$:     ENABL                   ;** Enable **
000126  004737  000000G                 CALL    GETMEM          ;See if there is enough memory for the job
000132  103403                          BCS     5$              ;Br if there is not enough free memory
                                ;
                                ;  There is enough free memory for the job.
                                ;  Swap it in,
                                ;
000134  004737  001250'                  CALL    INSWP           ;In swap the job
000140  000465                          BR      9$              ;Finished
                                ;
                                ;  There is not sufficient free memory space available for the job we want.
                                ;  See if there are some lower priority jobs that should be outswapped to
                                ;  make room.
                                ;
000142  010102                  5$:     MOV     R1,R2           ;Save # of job we want to bring in
                                ;
                                ;  Search job list from lowest priority job toward higher priorities looking
                                ;  for a job to outswap.
                                ;
000144                          10$:    DISABL                  ;** Disable **
000152  113701  000000G                 MOVB    RUNQTL,R1       ;Point to tail of job list
000156  001453                          BEQ     8$              ;Br if list empty
000160  020102                  11$:    CMP     R1,R2           ;Have we reached job we want to bring in?
000162  001451                          BEQ     8$              ;If yes, there is no lower prio job to swap
                                ;
                                ;  Found a lower priority job.
                                ;  See if it is in memory now.
                                ;
000164  032761  000000G 000000G         BIT     #$INCOR,LSW(R1) ;Is the job in memory now?
000172  001434                          BEQ     3$              ;Br if not
```

```
                                        ;
                                        ;  We cannot outswap a job if it is locked in memory
                                        ;
000174  032761  000000G 000000G              BIT     #$MLOCK,LSW6(R1);Is job locked in memory?
000202  001030                               BNE     3$              ;Br if yes
                                        ;
                                        ;  See if job has gotten minimum core residency time since last inswap.
                                        ;
000204  005761  000000G                      TST     LMINQ(R1)       ;Has min core residency time expired?
000210  001417                               BEQ     4$              ;Br if yes
                                        ;
                                        ;  Job has not yet gotten its minimum core residency time.
                                        ;  If job we want to inswap is in a high-priority real-time state,
                                        ;  ignore the minimum core residency time and do the inswap anyway.
                                        ;
000212  026227  000000G 000000G              CMP     LSTATE(R2),#S$$RT;Is job we want to inswap in real-time state?
000220  101413                               BLOS    4$              ;Br if yes
                                        ;
                                        ;  If job is waiting for something other than I/O, ignore min core time.
                                        ;
000222  016100  000000G                      MOV     LSTATE(R1),R0   ;Get current job state
000226  020027  000000G                      CMP     R0,#S$$RUN      ;Is job in an executable state?
000232  103403                               BLO     22$             ;Br if yes -- honor min core time
000234  020027  000000G                      CMP     R0,#S$IOWT      ;Is job in I/O wait state?
000240  001003                               BNE     4$              ;Br if not -- ignore min core time
                                        ;
                                        ;  We would like to outswap this job but its minimum core residency
                                        ;  time prevents the swap.  Set a flag that will cause the clock
                                        ;  interrupt routine to recall the scheduler when the minimum core time
                                        ;  expires.
                                        ;
000242  110137  000000G          22$:    MOVB    R1,SWPCOT       ;Set flag for clock interrupt routine
000246  000406                           BR      3$              ;Don't outswap job
                                        ;
                                        ;  We want to outswap this job.
                                        ;  If it has I/O active now, set flag to hold future I/O starts.
                                        ;
000250  105761  000000G          4$:     TSTB    LIOCNT(R1)      ;Does job have any I/O active now?
000254  001407                           BEQ     6$              ;Br if not -- swap it
000256  012761  000000G 000000G          MOV     #IOHLTM,LIOHLD(R1);Hold future I/O starts for job
                                        ;
                                        ;  Job we looked at is not eligible to be outswapped.
                                        ;  Try next job in list.
                                        ;
000264  116101  000001G          3$:     MOVB    LQLINK+1(R1),R1 ;Get # of next higher priority job
000270  001333                           BNE     11$             ;Br if one exists
000272  000405                           BR      8$
                                        ;
                                        ;  We have found a job to outswap.
                                        ;
000274                           6$:     ENABL                   ;** Enable **
000302  004737  000634'                  CALL    OUTSWP          ;Initiate the outswap
                                        ;
                                        ;  We've done all we can for now.
                                        ;
000306                           8$:     ENABL                   ;** Enable **
                                        ;
```

```
    115                                 ;  Finished
    116                                 ;
    117 000314  012602                  9$:     MOV     (SP)+,R2
    118 000316  012601                          MOV     (SP)+,R1
    119 000320  000207                          RETURN
```

```
                                        .SBTTL  SWPFUN -- Perform special swapping functions
                                ;-----------------------------------------------------------------------
                                ;  SWPFUN is called after any forced outswapping is completed to perform
                                ;  any special swapping functions such as locking jobs in memory or
                                ;  allocating memory for shared PLAS regions.
                                ;  Commands to SWPFUN are queued by use of swapper command packets.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> No special function in progress.
                                ;   C-flag set ==> Special function in progress,
                                ;                  don't start any other swapping activity.
                                ;
000322  010546                  SWPFUN: MOV     R5,-(SP)
                                ;
                                ;  Get pointer to 1st pending command packet
                                ;
000324  013705  000002'         2$:     MOV     SCPQHD,R5       ;Get pointer to 1st pending command packet
000330  001002                          BNE     1$              ;Br if there is a pending command
                                ;
                                ;  There is no pending command
                                ;
000332  000241                          CLC                     ;Say to continue normal swapping
000334  000405                          BR      9$
                                ;
                                ;  Call appropriate routine to process the command
                                ;
000336  116500  000000G         1$:     MOVB    SP$CMD(R5),R0   ;Get command value
000342  004770  000354'                 CALL    @SWPVEC(R0)     ;Call routine
                                ;
                                ;  The swapping action routines take care of freeing the command packet
                                ;  and return with the c-flag set or cleared.
                                ;  If the C-flag is cleared on return, go back and see if there is
                                ;  another command pending.
                                ;
000346  103366                          BCC     2$              ;Br if we should resume swapping activities
                                ;
                                ;  Finished
                                ;
000350  012605                  9$:     MOV     (SP)+,R5
000352  000207                          RETURN
                                ;
                                ;  Vector of addresses for swapping action routines
                                ;
000354  000360'                 SWPVEC: .WORD   SWPLOK           ;SA$LOK -- Lock job in low memory
000356  000440'                         .WORD   SWPRGN           ;SA$RGN -- Allocate shared region
```

```
                                        .SBTTL  SWPLOK -- Lock job in low memory area
                                ;-----------------------------------------------------------------------
                                ;  SWPLOK is a swapper action routine called to swap a job into a low
                                ;  memory region and lock the job in memory.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to swapper command packet.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Don't start any other swapping activities.
                                ;   C-flag cleared ==> OK to continue with normal swapping activities.
                                ;
000360  010146                  SWPLOK: MOV     R1,-(SP)
                                ;
                                ;  Get number of job that is to be locked in memory
                                ;
000362  116501  000000G                 MOVB    SP$JOB(R5),R1   ;Get # of job to lock in memory
                                ;
                                ;  Set flag saying job is locked in memory
                                ;
000366  052761  000000G 000000G         BIS     #$MLOCK,LSW6(R1);Say job is locked in memory
000374  042761  000000G 000000G         BIC     #$NLOCK,LSW6(R1);Clear needs-lock flag
                                ;
                                ;  Allocate memory space for the job.
                                ;  (Note, the memory space should be at the bottom of memory)
                                ;
000402  004737  000000G                 CALL    GETMEM          ;Get memory for the job
000406  103005                          BCC     1$              ;We should always branch
000410                                  DIE     #EM$LMF         ;Should never happen
                                ;
                                ;  Swap the job into the space we allocated
                                ;
000422  004737  001250'         1$:     CALL    INSWP           ;Inswap the job
                                ;
                                ;  Free the swap command packet
                                ;
000426  004737  001716'                 CALL    SCPFRE          ;Free the swap command packet
                                ;
                                ;  Finished
                                ;
000432  000261                          SEC                     ;Don't start any other swapping
000434  012601                          MOV     (SP)+,R1
000436  000207                          RETURN
```

```
                                        .SBTTL  SWPRGN -- Create shared global PLAS region
                                ;-----------------------------------------------------------------------
                                ;  SWPRGN is a swapper special function routine which allocates memory
                                ;  for a shared global PLAS region after swapping jobs out of
                                ;  memory to free up space for the region.
                                ;
                                ;  Inputs:
                                ;   R5 = Pointer to swapper command packet.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Don't start any other swapping activities.
                                ;   C-flag cleared ==> OK to continue with normal swapping activities.
                                ;
000440  010146                  SWPRGN: MOV     R1,-(SP)
000442  010246                          MOV     R2,-(SP)
                                ;
                                ;  Get a pointer to the Region Control Block out of the swap command packet
                                ;
000444  016501  000000G                 MOV     SP$DW1(R5),R1   ;Get pointer to Region Control Block
                                ;
                                ;  Try to allocate space for the region
                                ;
000450  016100  000000G                 MOV     RC$PAG(R1),R0   ;Get # 512-byte pages needed for region
000454  012761  177777  000000G         MOV     #-1,RC$BAS(R1)  ;Say address =-1 in case allocation fails
000462  004737  000000G                 CALL    TRYRGN          ;Try to allocate space for region
000466  103404                          BCS     1$              ;Br if cannot allocate (should never happen)
000470  072227  000003                  ASH     #3,R2           ;Convert page # to 64-byte block #
000474  010261  000000G                 MOV     R2,RC$BAS(R1)   ;Save pointer to region base in RCB
                                ;
                                ;  Make sure the job creating the region is in an executable state
                                ;
000500  116501  000000G         1$:     MOVB    SP$JOB(R5),R1   ;Get # of job that is creating region
000504  004737  000000G                 CALL    FORCEX          ;Make sure it is in executable state
                                ;
                                ;  Return the swap command packet
                                ;
000510  004737  001716'                 CALL    SCPFRE          ;Free the command packet
                                ;
                                ;  Finished
                                ;
000514  000241                          CLC                     ;Say normal swapping may continue
000516  012602                          MOV     (SP)+,R2
000520  012601                          MOV     (SP)+,R1
000522  000207                          RETURN
```

```
                                        .SBTTL  SWPMEM -- Do forced outswap of jobs
                                ;-----------------------------------------------------------------------
                                ;  SWPMEM is called to swap out of memory all jobs that have the $NDMEM
                                ;  flag set in LSW.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> All flagged jobs have been outswapped.
                                ;   C-flag set ==> Some jobs remain to be swapped.
                                ;
000524  010146                  SWPMEM: MOV     R1,-(SP)
000526  010246                          MOV     R2,-(SP)
                                ;
                                ;  Begin loop that checks each job to see if it needs to be outswapped
                                ;
000530  005002                          CLR     R2              ;Say not waiting on I/O to stop
000532  012701  000000G                 MOV     #LSTSL,R1       ;Get index # of last line
                                ;
                                ;  See if this job needs to be outswapped
                                ;
000536  032761  000000G 000000G 15$:    BIT     #$NDMEM,LSW(R1) ;Is this job waiting for memory expansion?
000544  001416                          BEQ     13$             ;Br if not
                                ;
                                ;  We found a job that needs to be outswapped.
                                ;  See if it is still in memory.
                                ;
000546  032761  000000G 000000G         BIT     #$INCOR,LSW(R1) ;Is this job still in memory?
000554  001412                          BEQ     13$             ;Br if not
                                ;
                                ;  The job is in memory, see if it has any I/O active.
                                ;
000556  105761  000000G                 TSTB    LIOCNT(R1)      ;Does this job have any I/O active?
000562  001003                          BNE     14$             ;Br if yes
                                ;
                                ;  Outswap this job.
                                ;
000564  004737  000634'                 CALL    OUTSWP          ;Start outswap on the job
000570  000411                          BR      9$
                                ;
                                ;  We cannot outswap this job because it has I/O active.
                                ;  Hold furture I/O starts for the job.
                                ;
000572  012761  000000G 000000G 14$:    MOV     #IOHLTM,LIOHLD(R1);Hold I/O for this job
000600  005202                          INC     R2              ;Remember we are waiting for I/O to stop
                                ;
                                ;  See if there are any other jobs that need to be outswapped.
                                ;
000602  162701  000002          13$:    SUB     #2,R1           ;Go check next job
000606  001353                          BNE     15$
000610  005702                          TST     R2              ;Do we need to wait for I/O to stop?
000612  001402                          BEQ     10$             ;Br if not
                                ;
                                ;  We have not finished all outswaps.
                                ;
000614  000261                  9$:     SEC                     ;Signal that outswapping still needed
000616  000403                          BR      11$
                                ;
                                ;  We have finished outswapping all flagged jobs
```

```
                                        ;
000620  005037  000000G                 10$:    CLR     MEMSWP          ;Say no more jobs to force out of memory
000624  000241                                    CLC                     ;Signal that all outswapping is finished
                                        ;
                                        ;  Finished
                                        ;
000626  012602                          11$:    MOV     (SP)+,R2
000630  012601                                  MOV     (SP)+,R1
000632  000207                                  RETURN
```

```
                                        .SBTTL  OUTSWP -- Outswap a job
                                ;-----------------------------------------------------------------------
                                ;  OUTSWP is called to swap a job out of memory.
                                ;  The job is immediately marked as not in memory.
                                ;  When the swap completes, the memory region used by the job
                                ;  is marked as free.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number of job to be outswapped.
                                ;   LBASE(R1) = Base page number allocated to job.
                                ;   LNBLKS(R1) = Number of pages allocated to job.
                                ;   LSWPBK(R1) = Swap file disk block # of swap area for job.
                                ;
                                ;  Outputs:
                                ;   SWPBSY = Number of job being outswapped.
                                ;   MEMMAP = Updated to show area used by job is free.
                                ;
000634  110137  000000G       OUTSWP: MOVB    R1,OUTBSY       ;Set flag saying outswap busy
000640  110137  000015'             MOVB    R1,LSTOUT       ;Save history of last job outswapped
000644  105037  000016'             CLRB    LSTOPR          ;Say outswap last thing done
                                ;
                                ;  Mark job as no longer in memory so scheduler won't try to run it
                                ;
000650  042761  000000G 000000G     BIC     #$INCOR,LSW(R1) ;Clear in-memory flag for the job
                                ;
                                ;  Indicate that memory mapping is not set up for this job
                                ;
000656  042761  000000G 000000G     BIC     #$MAPOK,LSW7(R1);Invalidate memory mapping data in contxt blk
000664  120137  000000G             CMPB    R1,MAPUSR       ;Is memory map currently loaded for this job?
000670  001002                     BNE     2$              ;Br if not
000672  105037  000000G             CLRB    MAPUSR          ;Say memory map no longer valid
                                ;
                                ;  See if we have to outswap any PLAS regions for the job
                                ;
000676  005761  000000G       2$:   TST     LNSBLK(R1)      ;Any PLAS regions associated with job?
000702  001404                     BEQ     1$              ;Br if no PLAS regions
000704                             OCALL   SEGOUT          ;Start outswap of PLAS regions
000712  103036                     BCC     9$              ;Br if PLAS outswap started
                                ;
                                ;  Select a slot in the swap file for this job
                                ;
000714  013702  000000G       1$:   MOV     VSWPSL,R2       ;Get # job slots in swap file
000720  013700  000000G             MOV     SWPJOB,R0       ;Point to table with job #'s in each slot
000724  005720               4$:   TST     (R0)+           ;Is this slot free?
000726  001406                     BEQ     3$              ;Br if yes
000730  077203                     SOB     R2,4$           ;Loop if more slots to check
000732                             DIE     #EM$SFO         ;Swap file overflow
000744  010140               3$:   MOV     R1,-(R0)        ;Claim swap file slot for our job
000746  163700  000000G             SUB     SWPJOB,R0       ;Calculate offset into table
000752  063700  000000G             ADD     SWPPOS,R0       ;Point into table with block #'s
000756  011061  000000G             MOV     (R0),LSWPBK(R1) ;Set block # in swap file for this job
                                ;
                                ;  Set up information to control swap of job root and context block
                                ;
000762  016137  000000G 000006'     MOV     LNBLKS(R1),SWPSIZ ;Number of 512-byte blocks to swap
000770  016137  000000G 000010'     MOV     LSWPBK(R1),SWPBLK ;Block number in swap file
000776  016137  000000G 000012'     MOV     LBASE(R1),SWPBAS  ;Base 512-byte block number in memory
```

```
     58 001004  004737  001012'                    CALL    OSWPGO          ;Initiate the outswap
     59                                 ;
     60                                 ;  Finished starting outswap
     61                                 ;
     62 001010  000207                  9$:        RETURN
```

```
                                        .SBTTL  OSWPGO -- Initiate outswap I/O operation
                                ;-------------------------------------------------------------------------
                                ;  OSWPGO is called to initiate an outswap I/O operation.
                                ;
                                ;  Inputs:
                                ;   SWPSIZ = Number of 512-byte pages to be written.
                                ;   SWPBAS = Starting page number within memory
                                ;   SWPBLK = Starting block number within swap file.
                                ;
001012  010146                  OSWPGO: MOV     R1,-(SP)
                                ;
                                ;  Get a system I/O queue element
                                ;
001014  012701  000000G                 MOV     #SWPCHN,R1      ;Point to system swap channel
001020  004737  000000G                 CALL    GETSYQ          ;Get a system I/O queue element (addr. in R1)
                                ;
                                ;  Set up I/O queue element for the transfer
                                ;
001024  063761  000010' 000000G         ADD     SWPBLK,Q.BLKN(R1) ;Set swap file block number for transfer
001032  013700  000012'                 MOV     SWPBAS,R0       ;Get base 512-byte block number within memory
001036  072027  000003                  ASH     #3,R0           ;Convert to 64-byte block number
001042  010061  000000G                 MOV     R0,Q.PAR(R1)    ;Set memory address in queue element
001046  013700  000006'                 MOV     SWPSIZ,R0       ;Get # 512-byte blocks to be written
001052  020027  000177                  CMP     R0,#127.        ;We can write a max of 127 per I/O operation
001056  101402                          BLOS    1$              ;Br if we can write all that is left
001060  012700  000177                  MOV     #127.,R0        ;Truncate write to 127 blocks this time
001064  160037  000006'         1$:     SUB     R0,SWPSIZ       ;Calc # 512-byte blocks left to transfer
001070  060037  000012'                 ADD     R0,SWPBAS       ;Calc memory page for next transfer
001074  060037  000010'                 ADD     R0,SWPBLK       ;Calc swap file block # for next transfer
001100  000300                          SWAB    R0              ;Convert # blocks to # words to write
001102  005400                          NEG     R0              ;Negative ==> Write operation
001104  010061  000000G                 MOV     R0,Q.WCNT(R1)   ;Set word count in I/O queue element
001110  012761  001126' 000000G         MOV     #OUTCMP,Q.COMP(R1) ;Set address of I/O completion routine
                                ;
                                ;  Queue the I/O request
                                ;
001116  004737  000000G                 CALL    SYQIO           ;Queue the I/O request
                                ;
                                ;  Return.  I/O completion will call OUTCMP.
                                ;
001122  012601                  9$:     MOV     (SP)+,R1
001124  000207                          RETURN
```

```
                                        ;-------------------------------------------------------------------------
                                        ;  Outswap I/O completion routine.
                                        ;
                                        ;  Check for swap file I/O errors
                                        ;
001126  032737  000000C 000000G OUTCMP: BIT     #CS$ERR!CS$EOF,SWPCHN ;Did any swap file I/O error occur?
001134  001405                         BEQ     3$              ;Br if not
001136                                  DIE     #EM$SIE         ;Abort if I/O error on swap
                                        ;
                                        ;  See if there is any part of the job remaining to be swapped
                                        ;
001150  005737  000006' 3$:     TST     SWPSIZ          ;Any part of job remaining to be swapped?
001154  001403                         BEQ     4$              ;Br if not
001156  004737  001012'                 CALL    OSWPGO          ;Initiate transfer of remainder of job
001162  000431                         BR      9$              ;We will be called again when transfer ends
                                        ;
                                        ;  Release the memory space occupied by the job just outswapped.
                                        ;
001164  113701  000000G 4$:     MOVB    OUTBSY,R1       ;Get job # being outswapped
001170  016102  000000G                 MOV     LBASE(R1),R2    ;Get base page #
001174  016100  000000G                 MOV     LNBLKS(R1),R0   ;Get # blocks assigned to job
001200  004737  000000G                 CALL    FREMEM          ;Release the memory space
001204  005061  000000G                 CLR     LBASE(R1)       ;Say job is no longer occuping memory
001210  042761  000000G 000000G        BIC     #$NDMEM,LSW(R1) ;Say job no longer needs outswap for mem expn.
                                        ;
                                        ;  If the job is in a suspended state waiting for memory expansion,
                                        ;  put in in an executable state now.
                                        ;
001216  026127  000000G 000000G         CMP     LSTATE(R1),#S$WFM ;Is job waiting for memory expansion?
001224  001004                         BNE     2$              ;Br if not
001226  012700  000000G                 MOV     #S$CPU,R0       ;Requeue job in run state
001232  004737  000000G                 CALL    ENQHD
                                        ;
                                        ;  Finished.  Request a job scheduler cycle.
                                        ;
001236  105037  000000G 2$:     CLRB    OUTBSY          ;Say outswap finished
001242  105237  000000G                 INCB    DOSCHD          ;Request a job scheduler cycle
001246  000207          9$:     RETURN
```

```
                                        .SBTTL  INSWP  -- Inswap a job
                                ;-----------------------------------------------------------------------
                                ;  INSWP is called to initiate the inswap of a job.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number of job to be inswapped.
                                ;   LBASE(R1) = Base page assigned to the job.
                                ;   LNBLKS(R1)= # 256-word pages currently used by job.
                                ;   LMEMIN(R1)= # 256-word pages to be assigned to job after inswap.
                                ;   LSWPBJ(R1)= Disk file block number where job is stored.
                                ;
                                ;  Outputs:
                                ;   LNBLKS(R1) is set to LMEMIN(R1) after swap finishes (used for
                                ;   memory expansion)
                                ;
001250  110137  000000G         INSWP:  MOVB    R1,INBSY        ;Say an inswap is in progress
001254  110137  000014'                 MOVB    R1,LSTIN        ;Remember last job inswapped
001260  112737  000001  000016'        MOVB    #1,LSTOPR       ;Remember inswap was last operation
                                ;
                                ;  If there are no blocks assigned to the job, then we are initializing
                                ;  the job and there is nothing to be read into memory.  In this case,
                                ;  call the completion routine.
                                ;
001266  005761  000000G                 TST     LNBLKS(R1)      ;Any blocks assigned to job?
001272  001500                          BEQ     INCMP2          ;Br if not -- Go call completion routine
                                ;
                                ;  We need to do an inswap for the job.
                                ;
                                ;  Set up information to describe the I/O transfer
                                ;
001274  016137  000000G 000006'         MOV     LNBLKS(R1),SWPSIZ ;Number of 512-byte blocks to read
001302  016137  000000G 000010'         MOV     LSWPBK(R1),SWPBLK ;Starting block number in swap file
001310  016137  000000G 000012'         MOV     LBASE(R1),SWPBAS  ;Starting page number in memory
                                ;
                                ;  Initiate the I/O operation
                                ;
001316  004737  001324'                 CALL    ISWPGO          ;Start the inswap
                                ;
                                ;  Finished
                                ;
001322  000207                          RETURN
```

```
                        .SBTTL  ISWPGO -- Initiate inswap I/O operation
                ;-----------------------------------------------------------------------
                ;  ISWPGO is called to initiate an inswap I/O operation.
                ;
                ;  Inputs:
                ;   SWPBAS = Base 512-byte page number of job in memory.
                ;   SWPSIZ = Number of 512-byte pages to read.
                ;   SWPBLK = Base block number in swap file.
                ;
001324  010146  ISWPGO: MOV     R1,-(SP)
                ;
                ;  Get a system I/O queue element
                ;
001326  012701  000000G         MOV     #SWPCHN,R1      ;Get address of swap file channel
001332  004737  000000G         CALL    GETSYQ          ;Get a system I/O queue element
                ;
                ;  Set up I/O queue element for the transfer
                ;
001336  063761  000010' 000000G         ADD     SWPBLK,Q.BLKN(R1) ;Set base block number in swap file
001344  013700  000012'                 MOV     SWPBAS,R0       ;Get memory page number
001350  072027  000003                  ASH     #3,R0           ;Convert to 64-byte page number
001354  010061  000000G                 MOV     R0,Q.PAR(R1)    ;Set memory address in I/O queue element
001360  013700  000006'                 MOV     SWPSIZ,R0       ;Get # 512-byte pages to read
001364  020027  000177                  CMP     R0,#127.        ;Can't read more than 127 per I/O operation
001370  101402                          BLOS    1$              ;Br if can read remainder of program this time
001372  012700  000177                  MOV     #127.,R0        ;Truncate and only read 127 this time
001376  160037  000006'         1$:     SUB     R0,SWPSIZ       ;Calc # blocks left to read next time
001402  060037  000012'                 ADD     R0,SWPBAS       ;Calc memory base address for next transfer
001406  060037  000010'                 ADD     R0,SWPBLK       ;Calc swap file block number for next transfer
001412  000300                          SWAB    R0              ;Cvt # blocks to # words to read
001414  010061  000000G                 MOV     R0,Q.WCNT(R1)   ;Set word count in I/O queue element
001420  012761  001436' 000000G         MOV     #INCMP,Q.COMP(R1);Set address of I/O completion routine
                ;
                ;  Queue the I/O request
                ;
001426  004737  000000G         CALL    SYQIO           ;Initiate the I/O operation
                ;
                ;  Return for now.
                ;  I/O completion will call INCMP.
                ;
001432  012601                  MOV     (SP)+,R1
001434  000207                  RETURN
```

```
                                ;-------------------------------------------------------------------------
                                ;  Inswap I/O completion routine.
                                ;
                                ;  Check for swap file I/O errors
                                ;
001436  032737  000000C 000000G INCMP:   BIT     #CS$ERR!CS$EOF,SWPCHN ;Did any swap file I/O error occur?
001444  001405                          BEQ     1$                ;Br if not
001446                                   DIE     #EM$SIE           ;Abort system if swap file I/O error
                                ;
                                ;  See if there is more of job root remaining to be inswapped.
                                ;
001460  005737  000006'         1$:     TST     SWPSIZ            ;Is any part of job remaining to be read?
001464  001403                          BEQ     INCMP2            ;Br if not
001466  004737  001324'                  CALL    ISWPGO            ;Start read of rest of job
001472  000207                          RETURN                    ;We will be called again when that I/O ends
                                ;
                                ;  Set # of pages in use by job in case it was outswapped due to a
                                ;  memory expansion request.
                                ;
001474  113701  000000G         INCMP2: MOVB    INBSY,R1          ;Get index number of job being inswapped
001500  016100  000000G                 MOV     LMEMIN(R1),R0     ;Get total # 512-byte pages used by job
001504  166100  000000G                 SUB     LNSBLK(R1),R0     ;Subtract # pages used by PLAS regions
001510  010061  000000G                 MOV     R0,LNBLKS(R1)     ;Set # pages used by root of job
                                ;
                                ;  See if we need to inswap any PLAS regions for this job
                                ;
001514  005761  000000G                 TST     LNSBLK(R1)        ;Any PLAS segments to inswap?
001520  001407                          BEQ     2$                ;Br if not
001522                                   OCALL   IASEGS            ;Set up memory addresses for PLAS regions
001530                                   OCALL   SEGIN             ;Start inswap of PLAS regions
001536  103004                          BCC     9$                ;Br if PLAS inswap started
                                ;
                                ;  Mark job as in memory now.
                                ;
001540  113701  000000G         2$:     MOVB    INBSY,R1          ;Get # of job being inswapped
001544  004737  001552'                  CALL    INSFIN            ;Finished inswap
001550  000207                  9$:     RETURN
```

```
                                        .SBTTL  INSFIN -- Finished inswap
                                ;-------------------------------------------------------------------------
                                ;  INSFIN is called when we have finished doing an inswap for a job.
                                ;
                                ;  Inputs:
                                ;   R1 = Index number of job that was inswapped.
                                ;
001552  010246                  INSFIN: MOV     R2,-(SP)
                                ;
                                ;  Say job is in memory now
                                ;
001554  052761  000000G 000000G        BIS     #$INCOR,LSW(R1) ;Say it is in memory now
                                ;
                                ;  Say slot in swap file is now free
                                ;
001562  013702  000000G                MOV     SWPJOB,R2       ;Point to table with job #'s in swap file
001566  013700  000000G                MOV     VSWPSL,R0       ;Get # slots in swap file
001572  020122                  1$:    CMP     R1,(R2)+        ;Search for job # in swap file table
001574  001402                         BEQ     2$              ;Br if found
001576  077003                         SOB     R0,1$           ;Loop if more to check
001600  000401                         BR      3$              ;Job must have been created without swapping
001602  005042                  2$:    CLR     -(R2)           ;Say swap file slot is free
                                ;
                                ;  Set minimum in-core run-time for job.
                                ;
001604  013761  000000G 000000G 3$:     MOV     VCORTM,LMINQ(R1);Must run this much before outswap
001612  005061  000000G                CLR     LQUAN(R1)       ;Initialize job's time quantum
                                ;
                                ;  Request a job scheduler cycle.
                                ;
001616  105037  000000G                CLRB    INBSY           ;Say inswap is finished
001622  105237  000000G                INCB    DOSCHD          ;Request a job scheduler cycle
                                ;
                                ;  Finished
                                ;
001626  012602                         MOV     (SP)+,R2
001630  000207                         RETURN
```

```
                                        .SBTTL  SCPGET -- Get a free swap command packet
                                ;-----------------------------------------------------------------
                                ;  SCPGET is called to get a free swap command packet.
                                ;  A system crash occurs if there are no free packets.
                                ;  The packet that is acquired is linked onto the queue of pending
                                ;  packets.
                                ;
                                ;  Outputs:
                                ;   R5 = Address of free packet.
                                ;
001632                          SCPGET:
                                ;
                                ;  Get 1st packet off of free list.
                                ;
001632  013705  000000G                 MOV     SCPFHD,R5       ;Get pointer to 1st free packet
001636  001422                          BEQ     10$             ;Br if there are no free packets
                                ;
                                ;  Remove packet from free list
                                ;
001640  016537  000000G 000000G         MOV     SP$LNK(R5),SCPFHD ;Remove packet from free list
                                ;
                                ;  Add packet to tail of pending list
                                ;
001646  013700  000004'                 MOV     SCPQTL,R0       ;Point to last entry in queue
001652  001403                          BEQ     1$              ;Br if queue is empty
001654  010560  000000G                 MOV     R5,SP$LNK(R0)   ;Make current end of tail point to us
001660  000402                          BR      2$
001662  010537  000002'          1$:    MOV     R5,SCPQHD       ;We are only entry in list
001666  010537  000004'          2$:    MOV     R5,SCPQTL       ;We are new end of list
001672  005065  000000G                 CLR     SP$LNK(R5)      ;Say no packet follows us
001676  105237  000000G                 INCB    DOSCHD          ;Request a job scheduler cycle
                                ;
                                ;  Finished
                                ;
001702  000207                          RETURN
                                ;
                                ;  Error -- There are no free swap command packets
                                ;
001704                          10$:    DIE     #EM$NSP         ;System crash -- No free swap command packets
```

```
      1                                                .SBTTL  SCPFRE -- Free a swap command packet
      2                                ;-----------------------------------------------------------------------
      3                                ;  SCPFRE is called to return a swap command packet to the free list.
      4                                ;
      5                                ;  Inputs:
      6                                ;   R5 = Pointer to packet to be freed.
      7                                ;
      8 001716                         SCPFRE:
      9                                ;
     10                                ;  Remove command packet from pending list
     11                                ;
     12 001716  016537  000000G 000002'       MOV     SP$LNK(R5),SCPQHD ;Make next packet be 1st pending one
     13 001724  001002                        BNE     1$                ;Br if there is another pending packet
     14 001726  005037  000004'               CLR     SCPQTL            ;No packets in queue
     15                                ;
     16                                ;  Add packet to free list
     17                                ;
     18 001732  013765  000000G 000000G 1$:   MOV     SCPFHD,SP$LNK(R5) ;Add packet to free list
     19 001740  010537  000000G               MOV     R5,SCPFHD
     20 001744  000207                        RETURN
     21         000001                        .END
Errors detected:  0

*** Assembler statistics

Work  file  reads:  0
Work  file writes:  0
Size of work file:  246 Words  ( 1 Pages)
Size of core pool:  18176 Words  ( 71 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:17.20
,LP:TSSWAP=DK:TSSWAP/C/N:SYM
```

Cross reference table (CREF V05.05)

```
$INCOR    1-25      3-18      3-56      7-26      8-24      14-12
$MAPOK    1-28      8-28
$MLOCK    1-26      3-61      5-21
$NDMEM    1-29      7-20     10-24
$NLOCK    1-33      5-22
CS$EOF    1-29     10-6      13-6
CS$ERR    1-31     10-6      13-6
DIEARG    1-43
DIEMSG    1-43      5-29*     8-47*    10-8*     13-8*     15-39*
DOSCHD    1-30     10-37*    14-32*    15-31*
EM$LMF    1-31      5-29
EM$NSP    1-32     15-39
EM$SFO    1-27      8-47
EM$SIE    1-26     10-8      13-8
ENQHD     1-28     10-32
FORCEX    1-33      6-33
FREMEM    1-24     10-22
GETMEM    1-28      3-29      5-27
GETSYQ    1-25      9-15     12-15
IASEGS    1-27     13-29
INBSY     1-23     11-16*    13-20     13-35     14-31*
INCMP    12-32     13-6#
INCMP2   11-25     13-13     13-20#
INSFIN    1-18     13-36     14-8#
INSWP     3-35      5-33     11-16#
INTPRI    1-26      3-28      3-108     3-113
IOHLTM    1-24      3-97      7-42
ISWPGO   11-36     12-10#    13-14
LBASE     1-25      8-57     10-20     10-23*    11-32
LIOCNT    1-29      3-95      7-31
LIOHLD    1-28      3-97*     7-42*
LMEMIN    1-27     13-21
LMINQ     1-27      3-66     14-26*
LNBLKS    1-30      8-55     10-21     11-24     11-30     13-23*
LNSBLK    1-26      8-35     13-22     13-27
LQLINK    1-25      3-20      3-102
LQUAN     1-25     14-27*
LSTATE    1-31      3-16      3-73      3-78      10-29
LSTIN     1-88#    11-17*
LSTOPR    1-90#     8-20*    11-18*
LSTOUT    1-89#     8-19*
LSTSL     1-27      7-16
LSW       1-27      3-18      3-56      7-20      7-26      8-24*     10-24*    14-12*
LSW6      1-24      3-61      5-21*     5-22*
LSW7      1-27      8-28*
LSWPBK    1-29      8-51*     8-56     11-31
MAPUSR    1-29      8-29      8-31*
MEMSWP    1-25      2-12      7-59*
OSWPGO    8-58      9-10#    10-14
OUTBSY    1-23      8-18*    10-19     10-36*
OUTCMP    9-33     10-6#
OUTSWP    1-18      3-109     7-36      8-18#
OVRHC     1-23      8-37     13-29     13-30
PSW       1-29      3-13*     3-28*     3-47*     3-108*    3-113*
Q.BLKN    1-30      9-19*    12-19*
Q.COMP    1-24      9-33*    12-32*
```

Cross reference table (CREF V05.05)

```
Q.PAR     1-30      9-22*     12-22*
Q.WCNT    1-29      9-32*     12-31*
RC$BAS    1-32      6-24*      6-28*
RC$PAG    1-32      6-23
RUNQHD    1-24      3-14
RUNQTL    1-30      3-48
S$$RT     1-28      3-73
S$$RUN    1-28      3-16       3-79
S$CPU     1-26     10-31
S$IOWT    1-30      3-81
S$WFM     1-33     10-29
SCPFHD    1-28     15-15      15-20*     16-18      16-19*
SCPFRE    5-37      6-37      16-8#
SCPGET    1-19     15-11#
SCPQHD    1-80#     4-17      15-28*     16-12*
SCPQTL    1-81#    15-24      15-29*     16-14*
SEGIN     1-27     13-30
SEGOUT    1-26      8-37
SP$CMD    1-31      4-27
SP$DW1    1-32      6-19
SP$JOB    1-31      5-17       6-32
SP$LNK    1-32     15-20      15-26*     15-30*     16-12      16-18*
SWAPER    1-18      2-8#
SWPBAS    1-84#     8-57*      9-20       9-28*     11-32*     12-20      12-28*
SWPBLK    1-83#     8-56*      9-19       9-29*     11-31*     12-19      12-29*
SWPCHN    1-25      9-14      10-6       12-14      13-6
SWPCOT    1-24      3-89*
SWPFUN    2-21      4-13#
SWPJOB    1-29      8-43       8-49      14-16
SWPLOK    4-44      5-13#
SWPMEM    2-14      7-10#
SWPPOS    1-31      8-50
SWPPRI    2-26      3-7#
SWPRGN    4-45      6-14#
SWPSIZ    1-82#     8-55*      9-23       9-27*     10-12      11-30*     12-23      12-27*     13-12
SWPVEC    4-28      4-44#
SYQIO     1-30      9-37      12-36
SYSHLT    1-43      5-29       8-47      10-8       13-8       15-39
TRYRGN    1-32      6-25
TSSWAP    1-6#      1-17
VCORTM    1-31     14-26
VSWPFL    1-24
VSWPSL    1-30      8-42      14-17
```

Cross reference table (CREF V05.05)

DIE	1-44#	5-29	8-47	10-8	13-8	15-39
DISABL	1-69#	3-13	3-47			
ENABL	1-73#	3-28	3-108	3-113		
OCALL	1-54#	8-37	13-29	13-30		